

CPU/CN



IN THIS ISSUE:

- Add 24 Commands to BASIC
- Wordpro 4 Review
- Buying a Printer
- BASIC 4.0 ROM Entry Points
- Beginning Machine Code
- Graph Plotting Package

**Volume 3
Issue 1**

 **commodore**

COMMODORE NEWS Andrew Goltz

British Software goes International

In the first three weeks since the announcement of the Commodore Management System series of business software a great deal of enthusiasm has been generated by these packages - not least from overseas.

Commodore representatives from several of our international companies attended a special software exhibition held by Commodore UK on the 16th and 17th September at the Skyway Hotel. Our British dealers expressed considerable interest in the packages which provide both 40 column and 80 column PET Users with a foundation of good quality business packages. Our overseas guests were impressed at the speed at which Commodore UK's Software Department had co-ordinated development effort on programs for the 8000 series PETs.

As a consequence "OZZ - The Information Wizard" has already been announced by Commodore USA, and it is now expected that both "OZZ" and "Stock Controller" will be on sale in the USA within the next few weeks. Other Commodore Software Managers have also expressed interest in the "Commodore Management System", not only in English-speaking countries like Canada and Australia, but also in other parts of the world where the programs will be modified to interact with a user in his own native tongue. French and German "translations" can be expected soon.

Me Too!

Not to be outdone by all this international dealing by Commodore UK, many Commodore Approved Products Suppliers used the occasion of the Skyway Software Exhibition to establish profitable overseas contracts. The top prize for business acumen must be handed to Paul Handover of Dataview. His popular Commodore Approved wordprocessing program "Wordcraft" so impressed Commodore International that they bought the overseas rights to "Wordcraft" lock, stock and print thimble. Consequently "Wordcraft", which will continue to be sold as "Commodore Approved" in the UK, will now be sold under the Commodore label in the rest of the world!

Wordpro joins Approved list

As part of a more aggressive marketing

plan "Wordpro" - for long a well-liked word processing program in Commodore's Business Software range, will now be sold directly by Professional Software as a Commodore Approved Product. This means that there will be two types of wordprocessing program, "Wordpro" and "Wordcraft" in the Commodore Approved Scheme. Both programs are high quality products with extensive wordprocessing features and comprehensive documentation. The difference lies in the way they tackle the problem of "user-interface", the way the edited text appears and is handled on PET's screen. There are advantages and disadvantages of both methods - and your choice will probably depend on your own taste as well as your intended application area. Ask your local dealer to demonstrate both programs and then choose the one that suits you best.

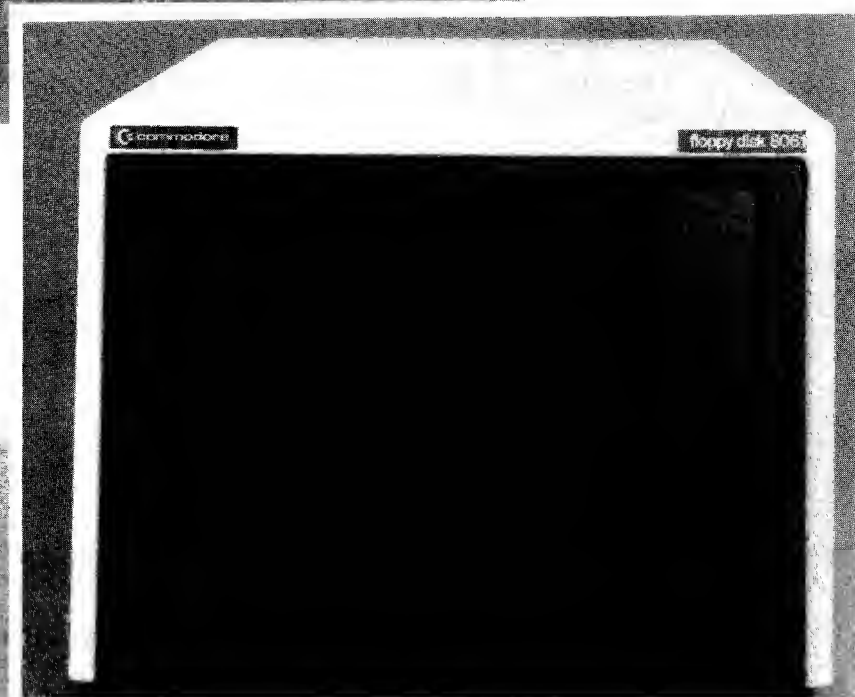
In the Labs

On September 30th, Commodore International announced a range of new products currently under development. Two products are of special significance for business applications.

The first of these is the 8096, a new 80 column, 96K memory PET which will be accompanied by the 8060 series disk drives. The 8060 series is a range of 8" drive products which will include the 8061, which can handle up to 1.6 megabytes and the 8062 which has the ability to store and access up to 3.2 megabytes of information. Both units have two vertical disk drives, but differ in that the 8061 utilises single sided drives, whilst the 8062 has double sided drive units. Especially significant is the fact that data is stored in IBM 3740 format which will make it very easy to exchange data with mainframe installations. If all runs smoothly, the 8096 "SuperPET" and the 8060 series drives will be with UK dealers in late spring 1981.

Colour PET will be called VIC

Also announced on September 30th was the VIC 20, a new colour machine aimed specifically at the education and home/hobbyist market. The machine utilises PET BASIC in ROM (making it a member of the PET family!) and comes with a full-sized keyboard and 5K of RAM (which can be expanded in stages up to 32k), colour, sound and high resolution graphics capability. VIC 20 will provide high quality colour pictures with an



ordinary colour domestic television set and utilises Commodore MOS Technology's latest Video Interface Chip to drive the composite video signal output. VIC 20, which is designed to be a user-friendly computer - "friendly in price, friendly in size, friendly to use and enjoy", is also expected to reach the UK in Spring 1981.

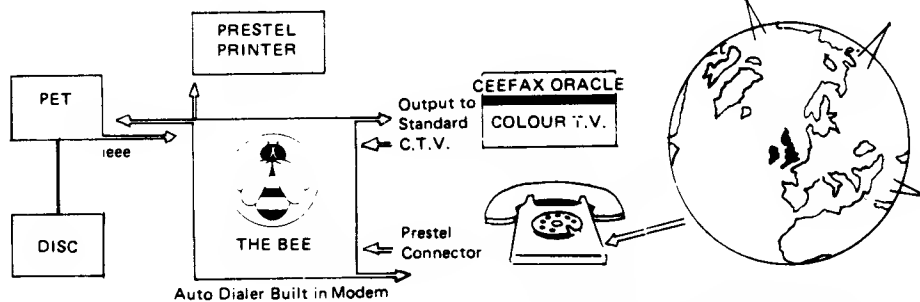
Keep Commodore Dealers Sane!

On the page opposite you will find photographs of the VIC as well as the 8096 and 8061. We must however stress that these are development machines in an advanced laboratory prototype stage and they are NOT currently available. Please do not phone up your local dealer, or Commodore for that matter and ask when they will be available because nobody knows as yet. We will notify you via CPUCN as soon as the new hardware arrives in this country for retail.

BEE LINES

FIRST PRODUCT FOR NATIONAL RELEASE

The Bee. (Prestel on your Pet)



£500 plus V.A.T.

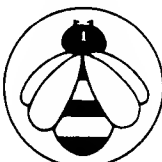
10% SECURES ONE OF FIRST DELIVERIES.

DEALER ENQUIRIES INVITED. PRESTEL IS A P.O. TRADE MARK

ALL CHEQUES TO B. & B. (COMPUTERS) LTD. RENTAL CAN BE ARRANGED.
DELIVERIES OF HARDWARE 60/90 DAYS. PENDING P.O. APPROVAL
SOFTWARE EX STOCK.

B&B [Computers] Limited

The Consultants for the North West



SUITE 1,
124 NEWPORT STREET,
BOLTON BL3 6AB.
LANCASHIRE.
Tel: (0204) 26644, 382741.



Skyway Software Exhibition

16th - 17th September



◀ Commodore's founder and Company President, Jack Tramiel.

(left to right) UK General Manager, Kit Spencer; Commodore's Chairman, Irving Gould and President Jack Tramiel. ▼



UK's answer to Jim Butterfield! Harry Broomhall ▼



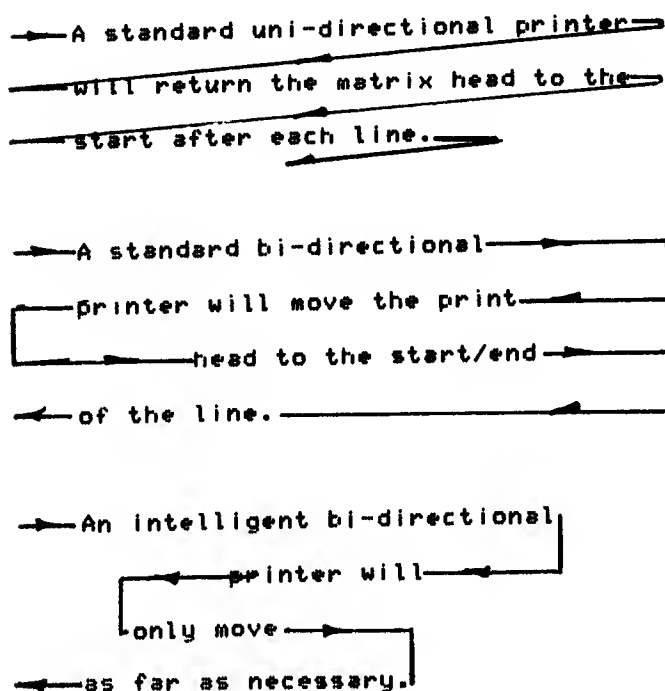
Always on the alert for KGB men — Digital Design and Development, one of the many stands at the exhibition.

A review of the 8024 Printer

The Commodore 8024 fast matrix printer is intended to complement the 8000 series computer for applications where hard copy is required in large quantities. The printer is very good looking with a sound absorbent see through perspex cover.

The 8024 is an intelligent bi-directional printer. This gives higher throughput than a uni-directional printer which returns the matrix head to the beginning of the line after printing. It is also faster than a normal bi-directional printer which will send the matrix head to the end of the current line when it has finished printing. By being intelligent the printer will only move the matrix head to the end or start of the next line of text rather than the physical edge of the print region. This can give dramatic speed increases when printing in the middle of the page eg.

Example of output from 8024



The printer also outputs the £ sign!

The printer has a 9x7 dot matrix which gives it good print definition and also the ability to print descenders with letters such as:- gjqy. The standard 7x7 dot matrix printer is incapable of doing this which can lead to some text being very difficult to read.

The maximum line width is 132 characters with 10 characters/inch but the 8024 has movable tractor feeds allowing any of the common size papers to be used including

labels.

The print speed is 160 characters/second at a sustained throughput which is far faster than most matrix printers available at present. A ribbon cartridge with an average life of 1 million characters makes ribbon changes both infrequent and also trouble free. Simply remove one and snap another into place.

There is an internal 132 character buffer which is filled before the line is printed. Characters can also be printed at double width for titles etc.

The 8024 has an internal bell which rings when the matrix head exceeds the edge of page sensors and also when the printer is about to run out of paper. The bell can also be made to ring under software control.

Up to 5 copies can be made simultaneously and there is a five position lever which allows the user to control the impact force of the matrix head.

The 8024 has switches for line feed and form feed. The paper can also be positioned by hand using a knob attached to the platen.

The printer responds to the same first 3 secondary addresses as used on the 3022/23 tractor printer so it is directly compatible with all current BASIC software using the 3022/23 printer under output format control but it will give the far faster speed and greater legibility which many businesses now require.

An innovation which will surprise most UK users is the provision for a pound sign. This is part of the character set in ROM and will replace the '#' symbol. This is an addition which many printers lack including the 3022/23 which normally has the pound sign as the user definable character. The 8024 has no user definable characters.

The Commodore 8024 costs £1160+VAT and will be available from your Commercial Systems dealer shortly.

The 8050 1 Megabyte disk system

The 8050 disk is part of the 8000 series hardware and has been designed for use with the 8032 SuperPET. The disk system is based around the well tried 3040 operating system which has proved to be very reliable.

Continued on Page 8

WordPro™

Word Processing Software

Everyone expected it would happen sooner or later... with WordPro it already has! Now all the marvelous benefits of an advanced stand-alone wordprocessor are available with the WordPro series of software and the systems they create.

If you've already been shopping for software in the crowded wordprocessing marketplace, you've probably determined the features you really want. You'll find WordPro has them, and more.

And if you haven't begun to shop yet, we urge you to compare, because only by comparison will you fully appreciate how complete and sophisticated WordPro software really is.

WHAT MAKES WORDPRO THE BEST?

Our research has shown that while many wordprocessing packages have comparable features to WordPro, none can surpass Wordpro's EASE OF USE AND FLEXIBILITY. Wordpro operators need not be familiar with computer commands or functions. WordPro is easy to learn for anyone with ordinary typing skills.

WORDPRO SOFTWARE IS LOADED WITH THE LATEST INNOVATIONS

Sophisticated systems programmed with leading edge wordprocessing features, WordPro is a series of programs designed specifically for use with the Commodore CBM/PET computers, peripherals and compatible typewriter quality printers.

WORDPRO SOFTWARE HELPS CREATE YOUR OFFICE OF THE FUTURE

With WordPro you'll institute remarkable new efficiencies and time savings into your office procedures. Letters and documents can be quickly processed, free of errors and corrections. Edits and last minute changes are no longer a matter of annoyance and lengthy retyping sessions. The document is simply recalled from the memory, edited on the screen and reissued in crisp, finished form.

Using WordPro, work that used to consume hours now takes minutes. WordPro's easy editing capabilities and virtually unlimited filing capacity frees your staff from repetitive typing and tedious filing tasks. This enables them to devote their time to more productive and rewarding duties. They will find the whole business of wordprocessing an exciting task. And your company will see the benefit of that excitement paid off in increased enthusiasm and productivity.

WORDPRO GIVES YOU THREE LEVELS TO CHOOSE FROM

WordPro software is a family of programs designed specifically for Commodore CBM computers, disk drives and compatible hardware (i.e. typewriter quality printers).



Turn your Commodore CBM/PET computer into a highly sophisticated word processing system

WordPro offers three levels of sophistication, from WordPro 1, which is a basic text editing system, to Wordpro 4, which brings you virtually every important feature offered by dedicated word processors. Each level of software is extremely easy to learn... and just as easy to operate! WordPro software brings you the very latest advances in high speed, efficient wordprocessing.



WORDPRO 1 is ideal for hobbyists, students and organizations who can benefit from the advantages of a basic wordprocessor without the program refinements of a commercially oriented system. WordPro 1 is recommended for use with the CBM/ PET 8/ 16K, C2N cassette and a properly interfaced printer.



WORDPRO 3 converts the CBM/ 2001 32K computer into a highly sophisticated 40-column screen wordprocessor. This program incorporates the advanced features considered important to effective wordprocessing, including nearly every entering, editing, memory and printing feature available today. WordPro 3 is recommended for use with CBM/ PET 32K (40-column) computer, CBM Dual Disk Drive, and a properly interfaced printer.

AVAILABLE FROM YOUR LOCAL COMMODORE DEALER. Join the hundreds of lawyers, educators, manufacturers, and businessmen who have already installed this remarkable software... and see for yourself.

WORDPRO 4 has it all! With this program, you will have everything you could want from a wordprocessor... and then some. WordPro 4 includes every feature found on WordPro 3, but with the added advantage of an 80-column display screen. The 80-column display simplifies text editing and makes entering text in columnar formats effortless. And with a few simple keystrokes, you'll be able to visualize on the screen exactly how your document will look prior to printing it out. WordPro 4 is designed for use with the Commodore CBM 8032 computer, CBM Dual Disk Drives, and a properly interfaced printer.

**Phone or write Professional Software for
the name of your nearest dealer
Professional Software**

**2nd Floor West End House
Hills Place
London W1
Telephone: 01-441 2397**

The outward appearance of the 8050 is nearly the same as the 3040 drive. Looking closer however it will be noticed that there is a small difference. The close down flap on the 3040 has been replaced with a different system. The disk is inserted into the drive and pushed until it clicks into place. The door is then pressed down. It will lock when fully depressed and the disk drive will auto-initialise. The disk is released by pressing the door down again slightly. To extract the disk from the drive the door is pushed in slightly.

The drives are made by Micropolis rather than Shugart as used in the 3040 and have two advantages. The first is the auto-initialisation mentioned above and the second is that the drive is self-aligning so there is no need to worry about the flap being left open for a few seconds to allow the disk to align as with the 3040.

A disk can theoretically hold an infinite amount of information but there are three limiting factors:-

1. The reliability of the disk. The floppy disk undergoes stringent tests to detect flaws in the magnetic surface. If the actual physical size of the recorded bit of information is large enough then there is little chance of disk flaws causing any loss of information.

2. The size of the read/write head limits the spacing between tracks. There can of course be no overlap between the tracks or previously recorded information will be lost.

3. The accuracy with which the read/write head can be positioned. There is little point having a system which can place data onto the disk very densely if the head reads back the data from the wrong track!

The new Micropolis drive, which has been used in many other manufacturers' disk systems has shown itself to be highly reliable. The read/write head is smaller and the drive which costs more than the Shugart has a greater accuracy. Point 1 is for the user to decide upon. With 500,000 bytes of data available purchase only the best quality disks.

The 8050 has some advanced features which only BASIC4 can access. The most useful of these is probably the relative record system which allows a very powerful random access data base to be set up without the user having to worry about which track and sector is being used, the relative record system takes care of all the housework. Another useful command is APPEND which will allow a sequential access file to be opened, data to be written to the end of the file and then closed without the file having to be read into PETs memory. The NEW command in DOS1 has been replaced by HEADER which includes a media check as the disk is formatted. BACKUP, which replaces

DUPLICATE, is three times faster, giving a backup disk in only two and a half minutes. CATALOG will give the same DIRECTORY readout but it is not loaded in memory, very useful for just checking to see if a file is on a disk when programming, without losing the program.

It should be noted that a disk produced on the 8050 can not be read by the 3040 and vice-versa. DOS2 as used in the 8050 is available as a retrofit for the 3040 drive and designated DOS2.1. The amount of storage is not increased but it will mean that programs such as OZZ can be run without having to upgrade the whole system. DOS1 can read disks produced on a DOS2.1 3040 but should not be used to write to the disk as it will corrupt it.

The cost of the 8050 disk system is £895.00+VAT and the DOS2.1 retrofit is £38.00+VAT. BASIC4 retrofit for the 3000 series PET costs £38.00+VAT.

Wordpro 4 — A review

WordPro is a program which will turn the PET into a powerful word processor. The program is designed to work on the 8000 series PET and is an ideal application for the 80 column screen allowing much more text to be displayed.

The keyboard of the 8000 series is much more suitable for the experienced typist, having a standard QWERTY layout and a full stop which is in the correct position at the lower right hand side of the keyboard, rather than being part of the numeric keypad. Cursor control and editing keys are grouped around the return key and the RUN/STOP key which was in the worst position on the keyboard on the 3000 series has been moved out of the way to the top of the keyboard.

WordPro 4 is virtually the same as the now well established WordPro 1, 2 and 3 with a couple of extra functions which virtually double the effectiveness of the system. These will be discussed later.

WordPro works with two sections of memory, one is called Main Text and holds the body of the letter, contract or page while the other section of memory is called Extra Text and is used to hold names or addresses which can be fed into main text, this will give the capability for writing standard letters. Once it is set up WordPro will write the letters with very little assistance from the operator. The time saving that this can give an overworked secretary typing the same letter again and again with only a name and address to change is quite amazing. Extra Text is also useful for another function, for example, say 'Commodore Approved Product' is going to appear many times throughout the text then a short mnemonic is used as a key to the string. When the command append is given the user has to type in the mnemonic and then as soon as return is

pressed the words are taken from Extra Text and put at the current cursor position. A similar command allows whole pre-defined paragraphs to be taken from extra text and put into Main Text.

WordPro uses an 'in-text' command system whereby a tick at the start of the line is used to denote the start of a command sequence and depending on the command either a colon, a semi-colon or a carriage return is used to denote the end. The commands are all logical, for example 'lm' is 'left margin', 'lf' is 'line feed' and 'ju' is 'right justify'. The commands are all two characters long and followed by a number thus allowing margin settings, text centring and justification to be altered at will.

WordPro 4 allows tabs to be set and cleared along the 80 column screen width allowing tables to be set out with considerable ease. As an aid to laying out tables a numeric mode is available which allows the cursor to be positioned at the right hand side of the column using TAB and then letting the user input the number, as the individual numbers are input the rest of number moves to the left rather than to the right as normal. This will give perfectly aligned columns of cash figures for example.

There are numerous editing features which give WordPro a power which is generally only found on much more expensive systems. There is a search and replace function which will look through the text either stored locally in memory or stored on disk and replaces it with another string. A hunt facility gives the user the option of just finding a string. Once the word has been found WordPro drops into edit mode to allow the user to make any changes.

Whole words and sentences can be deleted and by setting a range it is possible to move, delete, copy or transfer groups of lines to other points in the text. This is extremely powerful for text which is being written directly into the computer or for modifying an existing letter.

Disk storage is simple and effective. Text can be recalled so that it is attached to the end of the current text, or replaces the current text depending on the position of the cursor. A different command allows text to be inserted into the middle of that already resident in memory. All or part of the text may be saved to disk and if a file name is given and it already exists on the disk the option of replacing the file is given.

The main addition which WordPro 4 has over its other versions is in the output mode. The finished document can be sent

either to a the screen, the Commodore 3022 printer, a standard ASCII printer or to a Spinwriter. You will notice that this text is right justified but if you look you will also see that proportional spacing is being used. This gives a very high standard to the finished document as the text lacks the regular layout which is present with typewritten work. The Spinwriter gives the document a layout which can normally only be matched by typesetting. The disadvantage with using a Spinwriter or for that matter any printer is its output speed. WordPro 4 cures this by having an option which allows output to the screen of the PET. The document can then be read to see if the layout is correct.

Another addition which will prove to be useful is the pound sign. This is called up by redefining a key. For example, assume the '#' symbol is not going to be used. This will be printed to the PET screen but when it is sent to the printer it will cause the pound sign to be printed.

WordPro is very simple to get used to. Helen, Market Supports secretary mastered WordPro sufficiently well to be able to demonstrate its capabilities only three days after joining Commodore. There are a few minor criticisms which are more annoying than anything else. When reading the directory from disk the text in memory is erased. By switching into Extra Text 23 lines of the directory can be read, if there are more than 23 entries in the directory those at the end are omitted. A more serious complaint is in the splitting of words at the right hand end of the screen. It tends to be fairly difficult to read the text when the words get split and it can also lead to spelling mistakes and spaces being missed out.

In conclusion I would say that I have been using WordPro now for 3 months and have produced a great deal of text with it. The program turns the PET, especially the 8000 series into a very powerful word processor which anybody in business will find a use for, even if its only to write a standard begging letter to your bank manager asking for more money!

WordPro is a Commodore Approved Product and should be available through all Official Commodore Dealers. WordPro 3 for the 3000 series costs 275 pounds and WordPro 4 will cost 395 pounds.

Wordcraft 80, the other Commodore Approved Product wordprocessor will be reviewed in the next edition of CPUCN and a comparison made between the two.

BUSINESS USERS COLUMN

The Choice of a Printer

**Barry Miles,
Financial Consultant,
North London Polytechnic**

At first glance it might be imagined that the choice of a printer is easy. However as we shall discover this is not the case.

For instance, we may find the print from a daisy-wheel printer very easy to read but our real need may be for a printer which outputs very quickly, the output from which is relatively easy to read, whilst not being beautiful to look at.

Firstly we must determine what we want to do and what the criteria are by which we should judge a printer.

Individual business users will place different weighting on various attributes but the following items will need consideration.

To whom is the output to go? If it is for internal use only the quality of the printed output is not of prime importance, provided that the document is readable. If your letters are your shop window, in that they are your primary means of contact with your customers, then you attach far greater importance to their appearance than if you are producing invoices or statements.

Is speed of the essence or not?

If you have a large amount of printing to do, it will be false economy to have a relatively slow printer, just to save on capital outlay, when by doing so you tie up your entire computer system, thereby slowing down other important tasks and possibly keeping staff idle, who would otherwise be working productively for you.

Many readers will already have a printer and I suspect that in most cases it will be the Commodore tractor printer. There are many operations for which this machine is ideal. It is reasonably quick, legible and has the added advantage of being able to display the excellent graphics exactly as they appear on the screen. Some will say that in business applications graphics have no place but the use of boxes in particular will make sense of an otherwise almost incomprehensible array of information.

That said, the advantage of being able to print graphics is a minor one compared with the additional less-obvious one. In buying the Commodore computer system, you may well have been following the precept that it is sensible to restrict your choice of computer to one of the top-selling ones, thus ensuring that the manufacturer is likely to be around for a few years! Also knowing that the producers of the packaged programs, on which you are going to rely in order to obtain the maximum from your computer, will, in order to have the largest market for their wares, be likely to support only those machines which sell well.

Buying a popular machine ensures that you can look forward to being able to choose additional programs from a continuing array arriving from program designers. There is a considerable variety of printers on the market and sadly, standardisation has not taken place. Some programmers take this into account and their programs are designed to work with practically all printers; some will change programs to suit your printer, (for a fee) and some leave it to your dealer to modify the program. If the program is written in machine code, as many are, for reasons of enhanced speed, it is a far from trivial task to modify it to run satisfactorily on another machine. It follows that your safest bet is to buy a Commodore printer, since it will be fairly unlikely that a program designer would fail to support the manufacturer's printer.

Without getting too technical about this, the result of failing to adjust for a particular design of printer may be that everything is printed in double-spaced format, or that, worse still, the machine fails to carry out a line-feed operation every time a carriage return is executed, so that succeeding lines are printed on top of one another. This antic may be amusing when first encountered but is not recommended for continuing legibility.

You may find you need to print routine matters quickly but that you also have a need for good-looking, well-presented output. Your requirements may be best met by buying two machines: one the Commodore printer, the other a daisy-wheel.

For those not familiar with the term, a daisy-wheel printer is so called because the letters are arranged around the periphery of the wheel, like petals on a

PASCAL PROGRAMMING

Rob Evans

Following on from the REVIEW of Pascal in the last newsletter this article about Pascal is divided into three sections. The first, 'who will use Pascal', outlines exactly who I think will benefit from Pascal. The second section, 'how to learn Pascal', attempts to explain how you approach the problem of learning Pascal rather than actually trying to teach any syntax. This is because once you accept the structure of Pascal actually learning the syntax is relatively simple. The last section simply aims to get you familiar with PET Pascal.

Who Will Use Pascal?

There are many people who may consider using PET Pascal, including, the lecturer thinking of teaching Pascal to students; a writer for a software-house; or a hobbyist moving 'up market'. If you need reassurance of this look no further than the American Department of Defence (DOD) who are pouring oceans of money into developing ADA - a language which is based on Pascal!

The University lecturer thinking of teaching Pascal on the PET will no doubt be aware of the general advantages of teaching Pascal on a microprocessor over teaching Pascal (or any other language) on a mainframe-computer - which in a nutshell is 'user-friendliness'. As a student who has recently undergone both methods of learning a new language (ALGOL68 on a mainframe-computer and Pascal on the PET) I have no hesitation giving my vote everytime to the PET. We have sold 25 copies of Pascal to Strathclyde University who intend teaching Pascal to 250 first-years and more advanced features to computer science second-years. Strathclyde exhaustively tested Pascal before they invested so heavily in it and we regard their faith in PET Pascal as a tremendous compliment. Who knows they might even produce a 'Strathclyde Pascal', along the lines of 'Strathclyde BASIC'! Which as most of you will know is a training-kit comprising programs and manual. We have also had enquiries from UWIST, Cambridge University and Chichester College and I'm hoping my own University, Brunel, will use PET Pascal.

A software-house should weigh the advantages and disadvantages of Pascal to BASIC. Advantages are many and include the following:-

1. Programs which are easier to write

and debug because they are split into modules.

2. Programs which are easier to maintain because Pascal is easier to understand than BASIC.

3. Structured data types allowing, for example, lists and records.

4. Quicker program execution time (about 2 or 3 times quicker in the general case).

5. Immense satisfaction from writing in a structured-language.

6. Finally the hope that the more professional programmer will be attracted by Pascal.

No doubt the list could be greatly extended but I've listed what I consider the most important advantages. One disadvantage with Pascal vis-a-vis BASIC is that strings are not so well catered for and must be manipulated via arrays. Software-houses that write some routines in machine-code note that they can still enter these routines from within a Pascal program.

Turning to the hobbyist I admit 120 pounds may seem a lot to spend in one go, but this must be offset with the many hours of pleasure I hope you will receive and dare I say prestige associated with being a Pascal user. Incidentally for a new language this is not considered expensive, a view tacitly shared I'm sure by those reacting with a thoughtful nod, when we mentioned the price at recent exhibitions.

How to Learn Pascal

Anyone who already knows a block-structured language such as ALGOL or CORAL will find Pascal a delightfully simple language to learn. It is very straight-forward and neat. Because of this those approaching block-structured languages for the first time could not have chosen a better language and should experience none of the frustrations associated with similar languages such as ALGOL68, which is extremely complex.

For those outside the educational environment who have decided to teach themselves Pascal be, open-minded in your approach. The reason I mention this is that when switching from BASIC to Pascal one enters a new and strange programming world, familiarity with which will be richly rewarded with better and more satisfying programs.

If you are to teach yourself Pascal then

you need look no further than the manual supplied with your copy of Pascal. In my opinion it sets a new high standard for manuals. It contains a chapter for beginners to Pascal and has 101 pages. Useful are the numerous example programs included where new features of the language are introduced. As supplementary reading I can personally recommend 'Pascal User Manual and Report' by Jensen and Wirth, published by Springer-Verlag, which costs £5.95. After an initial, hopefully painless, period familiarising yourself with the concepts of block-structured languages the transition from BASIC to Pascal programming should be quite smooth and I am confident that in the long run the rewards, both in terms of professionalism and satisfaction, far outstrip the initial effort.

As a final note on learning Pascal I would hope that the Training Department will offer a course in the near future, so watch out for announcements.

Practical Experiences of Developing a Pascal

As an insight into program development I'll briefly explain the method I adopted developing a program to play a word game I know as Jotto or bulls-and-cows (with no bulls only cows!). The game is not unlike a word mastermind type game which uses words of 5 letters and the program knows 1926 words. I use a 1926x5 character array which indicates the size of array available in Pascal. After deciding the logic the program was divided into a main-logic section and a number of procedures. The main-logic therefore consists of Pascal statements and procedure calls. To facilitate communication between main-logic and procedures a common variable list was used. This list was created as a file (logically enough called variables!) and subsequently appended to the front of each individually compiled module i.e. main-logic and procedures. The appendage is achieved in one short line:

```
#filename
```

So if a variable changes in any way you only have to alter one set of variables and simply recompile each module.

To illustrate a Pascal program you might like to study the following Pascal source lines:-

```
10 program screenprint;
20 var i,j:integer;
30 begin
40   page;
50   for i:=0 to 24 do
60     for j:=0 to 39 do vdu(i,j,'*')
70 end.
```

The first thing to notice is the layout of the Pascal source lines. Not each line

starts in the first column. All Pascal programs are written in this fashion to reflect the structure of the program and the above program may equally have been written (remember it's on a 40 column PET):-

```
10 program screenprint;var i,j:integer;b
egin page; for i:=0 to 24 do for j:=0 to
20 39 do vdu(i,j,'*') end.
```

Let me explain. All lines of code starting in the same column will be executed the same number of times, and this number may be from 0 to infinity. To help you there are THREE types of loop in Pascal, one of these, the 'for' loop you will have used in BASIC. If lines of code starting in the same column (or blocks of code) are contained in other blocks then they will be repeated each time the higher block is repeated. So back to our example the 'begin' on line 30 and the 'end' on line 70 brackets our program block, all code within the 'begin' and 'end' is executed once. And reading from the top, 'page' is executed first, where 'page' is a handy Pascal function which clears the screen. Next, on line 50, we find the first 'for' loop and this is executed 25 times, however EACH time this loop is executed the subordinate 'for' loop is executed 40 times. The function 'vdu' writes the given character to the screen coordinate, so the first time it is executed '*' is written to the top left hand corner of the screen. The program therefore clears the screen and then fills the screen with '*' row at a time. Also you may have noticed that the two integer variables i and j had to be declared before use.

The advantage of compiling each module separately is that you can concentrate on logically independent and hopefully small sections of code at a time. These can be clean compiled before progressing to the next.

When a module is clean compiled Pascal produces a second file, of the same name, but with the suffix '.obj'. For example, a module called 'main' becomes 'main.obj'. This file, known as the object, contains the P-code for that module, and it is this P-code file that is subsequently interpreted at run time.

However before the program can be executed each constituent module must be clean compiled and the whole lot tied together by using the link command. The link command requires that you specify program name and constituent modules and outputs the modules as one big file which is thus labelled with the program name. This file is the one that is used when running the program.

By the way, to save typing effort store the link command as a file then simply recall this file and edit out the line number each time you execute a link. This is a good save as module-name's tend to be long and tedious to type. Also don't make link commands too long as they must

fit into 2 lines i.e. 80 columns. I made this mistake and found it a nuisance to recompile the modules with shorter names! Once the program has been successfully linked it can now be tested for logic errors i.e. debugged!

As a simple debugging aid displaying the procedure name upon entry into that procedure was extremely effective. This idea can be extended further by displaying pertinent variables at strategic places within procedures. Proceeding in this fashion bugs can be isolated to a particular module, which is amended, recompiled, and then relinked with the remaining modules before commencing the debugging procedure. For closer scrutiny of source code a printed listing can be obtained by setting a flag when compiling. Continuing in this fashion I found the program was developed reasonably quickly (You may even see the finished product if Commodore decide to release it!) and also with frequent

interruptions working in a busy office I found the method of dividing the program into modules extremely effective allowing one to concentrate on small sections of code.

Program User's Club

As I mentioned in the REVIEW of Pascal - see CPUCN vol. 2 no. 8 - we hope to be starting a Pascal User's Group, which would communicate via the CPUCN. Anyone interested in sharing their experiences of PET Pascal, whether they are programming tips or whatever should contact Dave Middleton at Commodore Slough.

MAKE MONEY IN THE MIDLANDS SOFTWARE DEVELOPMENT MANAGER

We are a leading micro-systems company based in the West Midlands specialising in the development of top quality application software for small/medium businesses.

We require a person (m/f) of exceptional ability to lead our software development team. Systems design experience, programming skills and familiarity with microcomputers are pre-requisites.

We offer a high basic salary and bonuses.

Please write in the first instance, with full details of experience, to:

**Mr. E. Hannah
214 Queslett Road
Great Barr
Birmingham**

All replies will be treated in the strictest confidence

£95 for a Commodore Approved Business Program?

**Save money by buying direct
from the software house and
receive full support from the
experts.**

**There are expensive alternatives but no other
record keeping program has these unique
capabilities:**

DMS has a DIRECT LINK TO WORDCRAFT

thus giving the opportunity to transfer information directly from DMS into standard letters.

DMS stores/prints information/letters/labels/does calculations.

DMS already has 250 users.

DMS is used for stock files/agency personnel/patient records to calculate VAT, sales figures, price changes, exchange rates, etc.

DMS is written by an established British software house.

DMS requires no training.

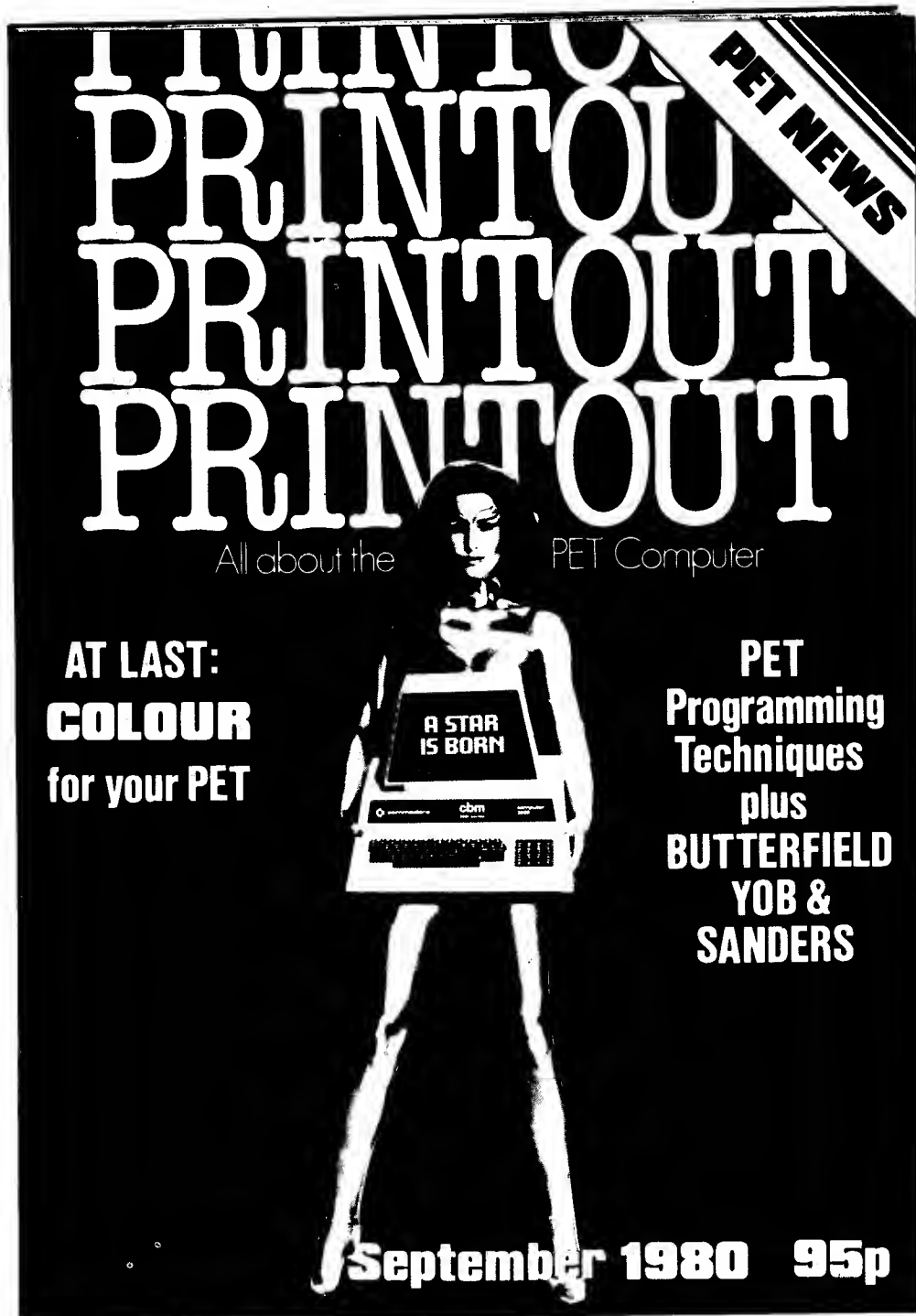
DMS is written in machine code — the fastest programming language.

Brochure on DMS on Commodore and CP/M from:

**Compsoft
Old Manor Lane
Chilworth
Guildford, Surrey
Telephone: 0483 39665**

PET SOFTWARE

— a guide



PRINTOUT — the independant magazine that's all about the CBM/PET computer.

Ten times a year PRINTOUT brings you the latest news about PET peripherals and software, conducts extensive and unbiased reviews, and tells you how to get the best out of your computer. In it you will find programming hints and listings — a complete Mailing List program free of charge in the latest issue — plus several fascinating pages of readers letters. There is even a gossip column! If you are interested in PET, you must read PRINTOUT. These are some of the features from the October issue:

- :: Profiles of Superchip and the Petaid database program
- :: 'What's wrong with WORDPRO', an evaluation of Commodore's word processor - and a guide to its use
- :: 'Memory from the Buffers' — How to get more memory
- :: 'Style & Technique' — How to write better PET programs
- :: 'Personal Electronic Transactions' by Gregory Yob
- :: Tommy's Tips — programming problems solved here
- :: 'Pets & Pieces' - our irrepressible columnist Gavin Sanders

plus News, Software Reviews, Letters, Gossip and Listings

PRINTOUT

PO Box 48, Newbury,
Berkshire, RG16 0BD
Telephone (0635)-201131

enclose

☐ £9.50 for one year's subscription (UK)

☐ £10.50 for one year's subscription (Eire)

☐ £14.50 for a year's subscription (overseas)

☐ 95p for a sample issue (UK & Eire only)

NAME

ADDRESS.....

POSTCODE.....



BASIC PROGRAMMING

How to get started in computing — Strathclyde University comes to the home of computer novices . . .

A totally new concept of computer programming education, initially carried out as an experiment at Strathclyde University, will provide BASIC laboratory programming courses with 50 Commodore PET computers for up to 1,125 students each year.

With the release of STRATHCLYDE BASIC, the benefits of this successful experiment are now available in a Teach-Yourself BASIC Programming Course which brings University education standards and University levels of proficiency to the homes of all first time computer users and any educational establishment involved in teaching BASIC.

Whether this "free standing" course is attended at Strathclyde University or followed on the basis of self-tuition, the teaching tools for STRATHCLYDE BASIC are the same: pencil and paper, STRATHCLYDE BASIC (two cassette programs and one workbook) and the Commodore PET computer.

The course is presented in 23 working sessions. Detailed explanations and instructions are alternately followed in the exercise book, and with the insertion of the cassette program, displayed on the PET screen. This method provides the student with solid programming knowledge and the confidence to convert theoretical knowledge into practice on the computer. At the beginning of each session, the student is encouraged to get down to pencil and paper exercises. Then the PET computer takes the beginner step-by-step through the actual computer programming while displaying instructions and results on the screen.

Working to a given average or maximum time table (which takes into account reading and working with the PET computer), will ensure the certain amount of discipline necessary to achieve results. Designed for the newcomer to computing, if followed during regular periods of study, the course should not exceed 10 weeks, although it can be mastered in as little as five days.

Computer Science Department at Strathclyde University and in charge of the initial experiment, is the author of STRATHCLYDE BASIC.

He explains the reasons for running an experimental "free standing" BASIC course at Strathclyde University: "Strathclyde University is a technological university and has one of the largest engineering schools in this country. For many years, our students have been taught programming in the traditional way. Whilst we were fully aware of the shortcomings of the conventional teaching methods, the introduction of interactive programming courses was an impractical project. The facilities provided came nowhere near to supporting the number of students involved. The cost of purchasing a terminal and connecting it to a central unit were not economically justifiable. With the introduction of Commodore's PET computer in 1978, our project of providing universal computer education for all students suddenly became feasible. After careful evaluation of several microcomputers we decided on the purchase of 30 Commodore PET computers... which we are now increasing to 50 units."

"The main objectives for our experiment were: Raising the standard of programming ability which the majority of the students can reach in a given time, whilst reducing the number of hours of personal involvement by the teaching staff."

"Over 170 students from different disciplines participated at this experiment of a "free standing" BASIC laboratory course. No formal lectures are held - it relies entirely on supervised self-tuition. Equipped with a workbook, containing some 23 units, students worked on time-tabled sessions and were able to test their newly acquired knowledge on any of the 30 PET computers available. The result was excellent. At the end of the course, over 94% of the students completed the test satisfactorily. Half of the remaining 6% who were prevented from taking the test by illness, have passed it on a later occasion. The failure rate of about 3% falls far below anything in our experience for first year classes and suggests that the new method of teaching is valid. We have had an enthusiastic response to the course and in some cases, students have actually expressed worries as to its academic value, because they find it 'fun'."

"Large scale programming courses can now

Professor Andrew Collin, Head of the

be held at a very low cost and assuming a life of 4 years for the 30 machines, two full time demonstrators or organizers, maintenance costs and interest on capital, the cost per student comes to approximately £15 per year, which even in today's difficult financial climate, becomes an economically viable project."

Professor Collin's advice to those wanting to achieve proficiency in BASIC by the end of his course is: "Be your own teacher. Don't jump to the next lesson unless you have fully understood and mastered the previous one".

STRATHCLYDE BASIC (cassettes and handbook) is available from Commodore dealers for £12.00

Finding the Duration of a Key Press

Robert Christmas

If a key is depressed while a program is running we can use its value in the program with GET A, but if GET A is executed a second time while the key is still held down it will return a 0 in A. If we wished to use the value of A to control, say, the frequency and duration of a musical note we would have to be able to detect how long the key is held down.

```
10 DEF FNK(X) = -(PEEK(515)=26)-2*(PEEK(515)=18)-3*(PEEK(515)=25)
```

or for New ROM PETs use the following:-

```
10 DEF FNK(X) = -(PEEK(151)=26)-2*(PEEK(151)=18)-3*(PEEK(151)=25)
```

will return a 0 unless keys 1, 2 or 3 are depressed. In those cases it will return 1, 2 or 3 for as long as the key is held down.

```
Test it with 20 PRINT " "; FNK (0)
30 GOTO 20
```

Clearly this definition of FNK is rather clumsy. It is used because it makes the operation of the function obvious. Using some logical operators it should be possible to detect all the numeric keys. Any suggestions?

If you wish to use the keys to control the program as above, but do not want to get a string of numbers printed when the program stops, then before the end of the program POKE the keyboard buffer pointer back to 0 with POKE525,0 (New ROM: POKE158,0).

Super Basic

David Simons

This utility adds an extra 24 commands to the normal BASIC. Great care must be used when they are in use, lack of care may result in the PET crashing. The commands are :-

```
@L -Loads screen from memory.
@S -Saves screen in memory.
@W -Swaps screen with memory.
@I -Inverts screen.
@D -Scrolls screen down a line.
@U -Scrolls screen up a line.
@H -Helps display text by displaying one screenful at a time, it then waits for an input. If you press space it will display another page of information and if the full-stop key is pressed it will return to "READY" mode. (note the full stop will be displayed.)
```

Also note that if you want the repeat key as well as the paginator you should enable the repeat key before enabling the paginator, it also must not be initialized if it is already enabled, to avoid any trouble you can type "@K:@H"

```
@F -Fills screen with a character.
@M -Swaps characters on screen.
@B -Draws a bar on a grid of 40 by 136, impossible ?? No, look at the graphics on the keyboard. This routine leaves room at the top for a title and room at the bottom for labels.
@C -Clears a bar made by above command.
@P -Makes a hard copy of the screen on the printer(inverse video will not come out).
@O -Sends a 2040/3040 command to disk.
@G -Loads a program from disk without the ",8".
@V -Verifies a program on the disk without the ",8".
@E -Exhibits a sequential file on the screen from the disk.
@N -Converts decimal to hex.
@T -Sends the PET to a place on screen for the next "print". It is advisable to send the cursor to home before usage of this command.
@GOTO -Computed goto allowing you to say go to a line which has a line number equal to the variable.
@AD -Defines a format.
@AA -Prints string in a format (a bit like print using).
@R -Enables the repeat key.
@K -Disables the repeat key and the paginator, this should be done before I/O activity (Eg LOAD & SAVE)
@J -Places the string that follows the 'J' on the screen, so that it is in the middle of the screen.
```

Syntax and Usage of Commands

These commands can be used just like normal BASIC commands but if used after a "THEN" a colon should be placed straight after the "THEN".

The commands not listed below or on the next page do NOT need any parameters and may be used as listed above.

@G

Requires a string expression, this means you can use a string variable (containing the program name) after the 'G' or the name of the program in quotes ("). Eg-

@G "1:DEMO"

Would load demo off disk 1 so would this :-

A\$="1:DEMO":@G A\$

@V

Requires the same type of expression as "@G" but it will verify the program, (if the program is OK then it will return with "?SYNTAX ERROR" if it is not you will need to press [RUN/STOP] to return to BASIC.

@B

Must be followed with the column of the bar plot, a comma and the height of the bar.

@C

Must be followed with the column to be cleared.

@M

Must be followed with the old character, a comma and the new character.

@F

Must be followed with the PEEK/POKE character that is required.

@N

Should be followed by a number which is greater than or equal to 0 and less than or equal to 63999.

@GOTO

Exactly the same as "@N".

@E

Has to be followed by a string expression which includes the drive, the name and also the ",S,R". Eg:-

@E "1:APPLE,S,R"

Would display the file "apple".

@AD

Has to be followed by a string expression. With this command the '\$' sign represents where the string that will be formatted will go. Letters are allowed, the string MUST be followed with the @ sign

@A@

Must be followed by a string expression, which has to be followed

by the @ sign (inside the string).
Example of the 2 commands above.

```
10 @AD "TIME:HOURS $$:MINS $$:SECS  
$$@"  
20 @A@ TI$+"@"  
30 PRINT  
40 GOTO 20
```

@O

Is to be followed with one of the disk commands enclosed in a string. The commands that may be used in the string are :-

Command abbreviation

SCRATCH	S
NEW	N
DUPLICATE	D
COPY	C
VALIDATE(*)	V
INITIALIZE	I

(*) In older disk manuals this was called VERIFY but it has been changed since then.
Example of use.

@O "I1"

Would initialize drive one but so would :-

A\$="I1":@O A\$

@T

Has to be followed by the line to where the PET is to be sent, a comma and the column

@J

Has to be followed by a string and the last character of the string must be a @ sign.

@P

Though there are no parameters with this command it is perhaps wise to mention a few things. Inverse characters on the screen will not appear on the printer as inverse, as inverse characters don't look too good. You should poke 634 with 2 if you want standard print but if you want it to come out in double size (the top line won't) you should type poke634,1. Run stop will NOT work while this command is working.

To Use the Commands

Simply type SYS 30529 enable and SYS 30513 to disable. The toolkit commands will work in most circumstances but sometimes the PET may jam.

On the 32k PET the commands start at \$7700 and end at \$7C00 and the screen is stored at \$7C00 to \$7F00. The first cassette buffer is used for string storage in the @P and @AD commands (note if you use the @P command after the @AD the format you defined will be lost). Odd places in the 2nd cassette buffer are also used, places used in 0 page include \$00,\$01,\$02,\$20,\$21,\$FF,\$FEand \$FD.

Using the Basic Loader Program

Type it in as if it was a normal BASIC program, once you have typed it in save it-DON'T run it! Once you have saved it run it and type the number for arming it. Try a normal BASIC command and then work your way through the Super BASIC commands. If at any time the PET jams or behaves wrongly reset the machine, load Super BASIC and check the data statements (this will take quite some time !!). If you do not feel up to typing it in I will send you a machine code loader (loads Super BASIC into high memory in just 1 second) which is a much shorter program than the BASIC loader (that means it will load faster off tape/disk) on cassette for just 5 pounds. I will also send you a demo on the other side of the cassette.

David Simons
19 Reddings
Welwyn Garden City
Herts

```

0 POKE53,118:POKE52,255:POKE49,118:
  POKE48,255
1000 PRINT"J":FORT=32768T032807:POKET,
  102:POKET+960,102:NEXT
1010 FORT=32768T033729STEP40:POKET,102:
  POKET-1,102:NEXT
1020 PRINT"###"_:PRINT"###
  SUPER BASIC " :A$="#####
  ###"
1030 PRINT"#####DAVID SIMONS.":
  PRINT"###"
1040 PRINT"#####USE ON NEW ROM":
  PRINT"###"
1050 PRINT"#####32K PETS ONLY.":PRINT"###
  ###"
1052 PRINT"#####LOADING":PRINTA$"____
  "
1070 PRINTA$"M TO ENABLE SYS30529":
  PRINTA$"M TO DISABLE SYS30513"
1100 FORT=30464T031743:READA:POKET,A:
  NEXT
1110 DATA201,78,208,3,76,240,123,201,84,
  208,3,76,24,121,201,137,240,3
1120 DATA76,3,206,32,139,204,32,210,214,
  32,176,199,76,118,0,165,1,208
1130 DATA1,96,76,42,121,208,3,76,240,12,
  3,76,24,121,121,234,234,169,230
1140 DATA133,112,169,119,133,113,169,20,
  8,133,114,96,169,76,133,112,169
1150 DATA78,133,113,169,119,133,114,96,
  76,193,123,234,234,234,160,0,177
1160 DATA119,201,64,240,3,76,118,0,230,
  119,208,2,230,120,177,119,230,119
1170 DATA208,2,230,120,201,85,208,3,76,
  121,120,201,68,208,3,76,79,120
1180 DATA201,70,208,3,76,30,120,201,83,
  208,3,76,182,119,201,76,208,3,76
1190 DATA214,119,201,87,208,3,76,246,11,
  9,201,73,208,3,76,56,120,201,68
1200 DATA208,3,76,78,120,201,77,208,3,7,
  6,160,120,201,65,208,3,76,199,120
1210 DATA76,60,121,24,162,128,160,0,132,
  33,134,34,165,34,233,3,133,1,169
1220 DATA0,133,0,177,33,145,0,200,208,2,
  49,232,224,132,208,232,96,24,162
1230 DATA128,160,0,132,33,134,34,165,34,
  233,3,133,1,169,0,133,0,177,0

```

```

1240 DATA145,33,200,208,249,232,224,132,
  208,232,96,24,162,128,160,0,132
1250 DATA33,134,34,165,34,233,3,133,1,1,
  69,0,133,0,177,33,133,2,177,0,145
1260 DATA33,165,2,145,0,200,208,241,232,
  224,132,208,224,96,32,117,214
1270 DATA134,0,160,0,132,32,162,128,165,
  0,134,33,145,32,200,208,251,232
1280 DATA224,132,208,244,96,162,128,160,
  0,132,1,134,2,177,1,73,128,145
1290 DATA1,200,208,247,232,224,132,208,
  240,96,160,40,132,34,160,0,132
1300 DATA32,160,191,162,131,134,33,134,
  35,177,32,145,34,136,192,255,208
1310 DATA247,202,224,127,208,238,160,39,
  169,32,145,32,136,16,251,76,118
1320 DATA0,162,0,134,1,162,40,134,32,16,
  2,128,134,2,134,33,160,0,177,32
1330 DATA145,1,200,208,249,232,224,132,
  208,238,169,32,162,40,157,199,131
1340 DATA202,208,250,96,32,117,214,134,
  1,32,204,214,134,0,160,0,132,32
1350 DATA162,128,165,0,134,33,177,32,19,
  7,1,208,4,165,0,145,32,200,208
1360 DATA243,232,224,132,208,236,96,177,
  119,230,119,208,2,230,120,201
1370 DATA64,240,23,32,253,244,134,1,133,
  2,160,0,177,1,153,122,2,201,64
1380 DATA208,1,96,200,76,220,120,32,253,
  244,134,218,132,219,160,0,185
1390 DATA122,2,201,64,208,1,96,201,36,2,
  08,12,177,218,201,64,240,245,32
1400 DATA210,255,200,208,232,32,210,255,
  200,198,218,208,2,198,219,76,243
1410 DATA120,169,0,133,136,169,128,133,
  197,32,117,214,134,1,32,204,214
1420 DATA134,198,198,1,24,165,196,105,4,
  0,133,196,165,197,105,0,133,197
1430 DATA76,33,119,201,79,208,26,169,1,
  133,210,169,111,133,211,169,8,133
1440 DATA212,32,253,244,32,36,245,169,1,
  133,210,32,172,242,96,201,82,208
1450 DATA54,169,103,133,144,169,121,76,
  247,122,165,151,201,255,240,10
1460 DATA197,255,240,9,133,255,169,16,1,
  33,254,76,46,230,165,254,240,4
1470 DATA198,254,208,245,198,253,208,24,
  1,169,4,133,253,169,0,133,151,169
1480 DATA2,133,168,208,227,201,75,208,9,
  169,46,133,144,169,230,76,247
1490 DATA122,201,80,240,3,76,53,122,76,
  24,122,169,128,160,0,132,33,133
1500 DATA34,162,0,177,33,32,16,122,201,
  64,48,36,201,126,48,37,201,128
1510 DATA48,28,201,191,48,34,201,255,24,
  0,35,157,123,2,232,224,40,240,32
1520 DATA200,208,220,230,34,165,34,201,
  132,208,212,96,105,64,76,206,121
1530 DATA105,128,76,206,121,233,64,76,2,
  06,121,169,191,76,206,121,169,13
1540 DATA157,123,2,232,169,0,157,123,2,
  132,0,169,122,160,2,32,28,202,164
1550 DATA0,200,76,179,121,201,128,48,2,
  233,128,96,234,169,4,133,210,169
1560 DATA111,133,211,169,4,133,212,162,
  4,32,36,245,162,4,32,201,255,32
1570 DATA171,121,32,231,255,96,201,72,2,
  08,21,165,144,141,82,122,165,145
1580 DATA141,83,122,169,122,133,145,169,
  84,133,144,76,118,0,76,133,122
1590 DATA76,0,0,165,196,201,192,48,6,16,
  5,197,201,131,16,3,76,81,122,173
1600 DATA18,232,201,191,240,12,201,255,
  240,245,169,147,32,210,255,76,81
1610 DATA122,169,230,133,145,169,46,133,
  144,169,0,133,158,76,137,195,201

```

```

1620 DATA67,240,3,76,187,122,32,117,214
,169,128,133,1,169,39,134,2,229
1630 DATA2,105,79,133,0,169,32,145,0,24
,165,0,105,40,133,0,165,1,105,0
1640 DATA133,1,201,131,208,235,165,0,20
,1,32,48,229,76,118,0,201,74,240
1650 DATA3,76,2,123,32,253,244,134,0,13
,3,1,32,86,214,169,40,134,2,229
1660 DATA2,133,2,169,32,32,210,255,198,
,2,198,2,165,2,201,1,16,241,160
1670 DATA0,177,0,201,64,208,1,96,32,210
,255,230,0,208,2,230,1,208,236
1680 DATA133,145,133,145,169,16,133,253
,76,118,0,201,69,240,3,76,82,123
1690 DATA169,2,133,210,133,211,169,8,13
,3,212,32,253,244,234,32,36,245
1700 DATA162,2,133,210,32,198,255,160,2
,32,207,255,201,13,208,3,76,58
1710 DATA123,201,0,208,3,76,231,255,32,
,210,255,76,33,123,32,210,255,165
1720 DATA151,201,255,240,3,76,231,255,1
,65,152,208,252,165,150,201,64,208
1730 DATA209,240,241,201,71,240,7,201,8
,6,240,7,208,27,0,160,0,240,2,160
1740 DATA1,132,157,162,0,134,150,134,20
,9,134,211,32,253,244,169,8,133
1750 DATA212,76,201,243,201,66,240,3,76
,237,123,169,130,133,1,32,117,214
1760 DATA134,2,169,247,229,2,133,0,32,2
,04,214,134,32,165,32,201,8,48,23
1770 DATA233,8,133,32,169,160,145,0,24,
,165,0,233,39,133,0,165,1,233,0
1780 DATA133,1,208,227,170,189,185,123,
,145,0,76,118,0,96,100,111,121,98
1790 DATA248,247,227,160,142,58,3,186,1
,89,1,1,201,249,208,16,189,2,1,197
1800 DATA198,240,9,230,119,208,2,230,12
,0,76,84,119,174,58,3,230,119,208
1810 DATA2,230,120,76,118,0,133,145,76,
,118,0,76,0,119,32,139,204,32,210
1820 DATA214,32,117,231,152,32,117,231,
,208,230,229

```

David Middleton

The version of Super BASIC shown here is obviously for the 32k PET BASIC2 machines but Dave has another version for a 16k PET BASIC2 so you will have to send him five pounds if you want to use the program, it is very good and some of the commands are very useful.

In CPUCN issue 2.8 David Pocock gave us a program which would give single key entry for the INPUT command. Not to be out done Dave Simons has sent me a program which will allow single key entry for any BASIC keyword, this will be published next time. Dave is also working on an article which will give details of how to interface new commands into the BASIC language. The technique is fairly simple but I have never seen it described before.

SOFTWARE PRIZE WINNER: DAVE SIMONS

It can hardly be a suprise that Dave Simons wins the Software Prize for the best article in this issue of CPUCN. Dave will get £50 of Commodore Software from the Master Library. Do not forget that there is also the chance of a £250

software prize which will be announced at the end of the volume.

```

10 FORA=0T039:0B A,2:NEXT B=1
20 FORI=0T0(39/B)STEP1/B:0B I*B,10+3*B
+4*B*SIN(I):NEXT B=B+1:IFB>20THEN
END
30 FORA=0T039:0C A:NEXT:GOTO20

5 FORA=0T039:0C A
10 FORA=0T039:0B A,2:NEXT B=1
20 FORI=0T0(39/B)STEP1/B:0B I*B,10+3*B
+4*B*SIN(I)
25 NEXT B=B+1:IFB>20GOTO5
30 FORA=0T039:0C A:NEXT:GOTO20

1 B=1
5 FORA=0T039:0C A
10 FORA=0T039:0B A,0:NEXT
20 FORI=0T0(39/B)STEP1/B:GOSUB100:0B I
*B,Y:0M 160,32
25 NEXT B=B+1:IFB>15THEN1
30 FORA=0T039:0C A:NEXT:GOTO10
100 Y=INT(4*B+4*B*SIN(I))
110 IFINT((Y+1)/8)=(Y+1)/8THENY=Y+1
120 RETURN

```

A program to plot graphs

David Middleton

The program I am about to describe does much more than simply plot graphs. It will take raw data from an experiment and give a double density plot of the results on scaled and labeled axes. The program was written for use in schools and colleges where an experiment is going to be performed many times to different groups of students. The program may find applications outside schools because once the method for altering the program have been mastered it is fairly quick to set it up for other examples.

```

1 REM
2 REM
5 REM ***PROGRAMME FLOW CONTROLLER***
6 REM
7 REM
50 GOSUB9000: REM SET UP DATA ARRAYS
55 GOSUB6000: REM INPUT VARIABLES
60 GOSUB1000: REM INPUT DATA
70 GOSUB2000: REM CALCULATE POINTS
80 GOSUB4000: REM SCALE X-Y
90 GOSUB5000: REM SET UP AXIS AND PLOT
100 END
500 REM
510 REM
1000 REM *** MAJOR INPUT MODULE ***
1001 REM
1002 REM
1120 FOR CO=1 TO NI
1130 PRINTNA$(CO): FOR DC=1 TO ND:
INPUTT(CO,DC): NEXTDC:
1140 PRINT:INPUT"IS THIS CORRECT Y/N###
Y####":AN$:IFAN$="N"GOTO1130
1150 PRINT: NEXT CO:
1200 CL=1
1210 FORY=1TONI:T(Y,0)=T(Y,CL):NEXTY
1220 LO=T(1,CL):CO=CL:FORX=CLTOND

```



```

1230 IFT(1,X)<LO THEN LO=T(1,X):CO=X
1240 NEXT X
1250 IFCO=CL GOTO 1270
1260 FOR Y=1 TO NI: T(Y,CL)=T(Y,CO):T(Y,CO)
      =T(Y,0):NEXT Y
1270 CL=CL+1:IF CL<ND GOTO 1210
1280 RETURN
1810 REM
1820 REM
2000 REM **** CALCULATION MODULE ****
2001 REM
2002 REM
2005 PRINT"UNCORRECTED MASS FLOW RATE M
      C, PRESS.RATIO P2/P1 & EFFICIENCY
      NT"
2010 PRINT" MC"," P2/P1"," NT":
      PRINT
2015 FOR CO=1 TO ND
2020 P=T(1,CO): S=T(2,CO): U=T(3,CO): V
      =T(4,CO): R=T(5,CO)
2025 Z=1-P/(13.6*M)
2030 W=(Z+1.428)-(Z+1.714)
2040 K=SQR(3.5*W)
2050 M=1.4113*K
2070 L=(U+H)/(H-S)
2090 G=(V+273)*(L+1.2857-1)/(R-V)
2100 PRINT M;L;G
2110 T(6,CO)=M:T(7,CO)=L:T(8,CO)=G
2120 PRINT
2130 NEXT CO
2135 PRINT"PRESS THE SPACE BAR TO CONTI
      NUE"
2140 GET A$:IF A$<>" "GOTO 2140
2150 RETURN
3000 REM
3002 REM
3005 REM *** PLOTTING MODULE ****
3006 REM
3007 REM
3200 FOR CO=1 TO ND-1
3210 X1=T(XP,CO): X2=T(XP,CO+1): Y1=T(Y
      P,CO): Y2=T(YP,CO+1)
3235 IF X1-X2=0 THEN MG=1:GOTO 3350
3240 MG=(Y1-Y2)/(X1-X2)
3250 CN=Y1-X1*MG
3253 IF X2>X1 GOTO 3290
3259 FOR X3=X2 TO X1 STEP 1: Y3=MG*X3+CN:
      GOSUB 3900:NEXT X3:GOTO 3400
3290 FOR X3=X1 TO X2 STEP 1: Y3=MG*X3+CN:
      GOSUB 3900:NEXT X3:GOTO 3400
3350 IF Y1<Y2 GOTO 3370
3360 FOR Y3=Y2 TO Y1: X3=X1:GOSUB 3900:NEXT Y
      3:GOTO 3400
3370 FOR Y3=Y1 TO Y2: X3=X1:GOSUB 3900:NEXT Y
      3
3400 NEXT CO:RETURN
3900 POKE 179,Y3:POKE 180,X3:SYS 847:
      RETURN
4000 REM
4001 REM
4002 REM *** SCALING MODULE ****
4003 REM
4004 REM
4010 FOR DC=NI+1 TO NI+OD
4020 HI=T(DC,1): LO=T(DC,1): FOR CO=2
      TO ND: IF T(DC,CO)>HI THEN HI=T(DC,
      CO)
4030 IF T(DC,CO)<LO THEN LO=T(DC,CO)
4040 NEXT CO
4050 IF MID$(GS$,DC-NI,1)="Y" GOTO 4100
4060 SX=75/(HI-LO)
4070 FOR CO=1 TO ND: T(DC,CO)=INT((T(DC
      ,CO)-LO)*SX)+3: NEXT CO:GOTO 4200
4100 SY=45/(HI-LO)
4110 FOR CO=1 TO ND: T(DC,CO)=INT((T(DC
      ,CO)-LO)*SY)+3: NEXT CO
4200 SA(DC-NI,1)=HI: SA(DC-NI,2)=LO:

```

```

      NEXT DC: RETURN
5000 REM
5010 REM
5020 REM *** PLOT AXIS MODULE ****
5030 REM
5040 REM
5050 PRINT"WHEN THE GRAPH HAS BEEN PLO
      TTED PRESS THE SPACE KEY TO CONT
      INUE"
5060 FOR CO=1 TO 1500: NEXT
5070 PRINT" ": FOR CO=1 TO 25: PRINT" ":
      NEXT
5080 PRINT" ";
5100 PRINT" "; CO=20-LEN(PT$(1+GC))/2:
      GOSUB 5900:PRINT PT$(1+GC)
5110 CO=8-LEN(PT$(2+GC))/2:GOSUB 5950:
      FOR CO=1 TO LEN(PT$(2+GC))
5115 PRINT MID$(PT$(2+GC),CO,1):NEXT
5120 CO=23:GOSUB 5950:CO=20-LEN(PT$(3+GC
      ))/2:GOSUB 5900:PRINT PT$(3+GC):
5125 GOSUB 7000
5130 PRINT" "; VA=SA(YP-NI,1):GOSUB 5800
      :PRINT VA$;
5135 CO=16:GOSUB 5950:VA=SA(YP-NI,2):
      GOSUB 5800:FOR CO=1 TO LEN(VA$)
5136 PRINT MID$(VA$,CO,1):NEXT
5140 CO=23:GOSUB 5950:CO=3:GOSUB 5900:VA
      =SA(XP-NI,2):GOSUB 5800:PRINT VA$;
5150 CO=23:GOSUB 5950:CO=30:GOSUB 5900:VA
      =SA(XP-NI,1):GOSUB 5800:PRINT VA$;
5200 PRINT" ";GOSUB 3000:GC=GC+3:WAIT 594
      10,4,4:IF GC<NG GOTO 5000
5210 RETURN
5800 REM
5801 REM
5804 REM THIS SUBROUTINE MAKES EVERY
      NUMBER CORRESPOND TO THE FORM X.X
      XE XX
5805 IF VA=0 THEN VA$="0.0":GOTO 5890
5810 VA$=STR$(VA*1.00000001E20):IF
      LEFT$(VA$,1)="-" GOTO 5830
5820 VA$=RIGHT$(VA$,LEN(VA$)-1)
5830 CO=VAL(RIGHT$(VA$,2))-20
5832 VA$=LEFT$(VA$,LEN(VA$)-3)+RIGHT$(
      STR$(CO),2)
5850 IF LEFT$(VA$,1)="-" THEN VA$=LEFT$(VA
      $,4)+RIGHT$(VA$,3):GOTO 5890
5860 VA$=LEFT$(VA$,4)+RIGHT$(VA$,3)
5890 RETURN
5895 REM
5896 REM
5900 FOR DC=1 TO CO:PRINT" ";:NEXT:RETURN
5950 PRINT" ";FOR DC=1 TO CO:PRINT" ";:
      NEXT:RETURN
6000 REM
6002 REM
6003 REM *** MINOR INPUT MODULE ****
6005 REM
6025 REM
6040 INPUT"NUMBER OF SETS OF DATA";ND:
      DIM T(NI+OD,ND), SA(OD,2)
6050 INPUT"ATMOSPHERIC PRESSURE MM OF H
      G";H
6100 RETURN
7000 REM
7001 REM
7002 REM *** PLOT CONTROLLER ****
7003 REM
7004 REM
7010 ON INT(GC/3+1)GOTO 7100,7200,7300
7020 PRINT"ERROR IN PLOT CONTROLLER":
      STOP
7100 REM 1ST GRAPH
7105 XP=6
7110 YP=7
7120 RETURN

```

```

7200 REM 2ND GRAPH
7205 XP=6
7210 YP=8
7220 RETURN
7300 STOP
8590 RETURN
9000 REM
9002 REM
9005 REM ***      DATA MODULE      ***
9006 REM
9007 REM
9010 REM INITIALISE DATA FOR QUESTIONS
9015 REM
9020 READ NI: FOR CO=1 TO NI: READ NA$(CO):NA$(CO)=NA$(CO)+CHR$(13):
NEXT CO
9025 READCH$:IFCH$<>"END"THENPRINT"ERR
R IN DATA MODULE PLEASE CHECK":
STOP
9030 DATA 5,PRESSURE DROP ACROSS NOZZLE
(MM H2O),INLET PRESSURE(MM HG)
9040 DATA OUTLET PRESSUE H2 (MM HG),INL
ET TEMP T01 (CENT.)
9050 DATA OUTLET TEMP T02 (CENT.),END
9060 REM
9070 REM
9100 REM GRAPH TITLES
9110 REM READ FROM DATA IN THE FORM MAI
NTITLE, X AXIS TITLE, Y AXIS TITLE

9120 READ NG: NG=NG*3: DIM PT$(NG):FOR
CO=1 TO NG: READPT$(CO):NEXT
9130 READCH$: IFCH$<>"END"THENPRINT"ERR
OR IN DATA MODULE PLEASE CHECK":
STOP
9140 DATA 2,MASS FLOW / PRESSURE,PRESSU
RE,MASS FLOW
9160 DATA MASS FLOW / EFFICIENCY,EFFICI
ENCY,MASS FLOW,END
9170 REM
9180 REM
9220 REM NUMBER OF SETS OF OUTPUT DATA
9230 OD=3
9240 REM
9250 REM
9320 REM GRAPH SCALE VARIABLE
9330 GS$="XYY"
9340 REM
9900 REM
9905 REM THIS COMES LAST AS IT IS ONLY
USED ONCE WHEN THE PROGRAMME IS L
OADED
9910 REM TO SET UP MACHINE CODE PLOT
ROUTINE.
9920 IFPEEK(826)=200GOTO9960
9930 LO=846
9940 LO=LO+1: READ PO: IF PO=999 GOTO99
60
9950 POKE LO,PO: GOTO9940
9960 LO=847: POKE826,200: RETURN
9970 REM
9980 REM DATA FOR M/C PLOT ROUTINE
9990 REM
10000 DATA165,180,74,133,178,144,5,169,
3,76,93,3,169,12,133,181,165,179,
74,133
10010 DATA177,144,7,165,181,41,6,76,113
,3,165,181,41,9,133,181,169,24,56
,229
10020 DATA177,48,35,133,177,56,233,25,1
6,35,169,128,133,180,169,0,133,17
9,164
10030 DATA177,240,14,105,40,144,5,230,1
80,24,234,234,136,208,244,133,179
,165
10040 DATA178,48,73,133,178,56,233,40,1
6,66,165,179,101,178,144,2,230,18

```

```

0,133
10050 DATA179,160,0,162,15,234,234,234,
177,179,168,221,239,3,240,5,202,1
6,248
10060 DATA162,0,138,5,181,76,204,3,69,1
81,170,234,234,234,189,239,3,162,
0,129
10070 DATA179,169,0,96,120,210,255,0,25
5,0,255,0,255,0,255,0,255,0,96,23
9,239
10080 DATA239,239,239,239,32,108,124,22
5,126,127,226,251,123,98,255,254,
97,252
10090 DATA236,160,999

```

The program shown is set up for processing the data produced from an experiment to determine the mass flow rate/efficiency and the mass flow rate/pressure of a compressor. This program will give the user five prompting questions to which the user has to give a pre-determined number of answers. The data does not have to be input in a sorted form, the program will sort the data by column when the input has finished. Note that when changing the program the sort routine works with the data input in response to the first question. Ensure that you choose the first question to have a response which will not produce a constant value eg. If you ask for the water inlet temperature you will get a response wich will be very similar: 24 degrees.

The program is very modular and only a few of the modules need to be altered to run data from different experiments so rather than deal with how the program works I will give instructions for changing it.

Firstly decide what data you are going to need in the experiment. As an example I am going to plot the graphs of two functions with two variables A and B with a constant C.

The two functions are:-
 $Y=X*X+Z$ where $X=A/B$ and
 $Y=SQR(ABS(X*X+Z))$ where $X=A+B$

DATA MODULE subroutine 9000

The data module determines the framework for the data base to be used in the program.

Lines 9010-9070 ask the questions and wait for the users response. The first item to change is in line 9030. The number gives the number of questions which are going to be asked. There are two sets of data required from the user so change the line to read:-

9030 DATA 2,INPUT A, INPUT B, END

Make sure you put END as the last item otherwise the program will stop. This gives a built in error check to ensure

that the program is being altered correctly.

Lines 9100-9180 are the graph titles. These are input in the order Main title, Y Axis and then X Axis, contrary to the REM statement in line 9110! The number of graphs to be plotted is entered as the first item of data eg.

```
9140 DATA 2,Y=X*X+Z,Y AXIS,X=A/B
9150 DATA Y=SQR(ABS(X*X+Z)),Y AXIS,X=A+B
```

Lines 9200+ give the number of sets of output data. There are going to be four sets of output data so OD=4. When the data is going to be scaled it needs to know which axis the data is going to be plotted on. Hence G\$="XYXY"

The rest of the data in this section is for the double density plot routine.

PLOT CONTROLLER MODULE subroutine 7000

The data for the program is held in a two dimensional array. The first two rows in the T array will be taken up by A in row 1 and B in row 2. Rows 3-6 are going to hold the output data in the form given in G\$ and for the graph titles. So now:-

```
7105 XP=3
7110 YP=4
7205 XP=5
7210 YP=6
```

MINOR INPUT MODULE subroutine 6000

The next change to be made is to get the constants for the program to be input. Line 6040 is very important and should be left alone. Lines 6050+ can be altered to give any constants used in the experiment eg.

```
6050 INPUT"INPUT THE CONSTANT C";Z
```

CALCULATION MODULE subroutine 2000

The calculation module is the most difficult routine to change as you are actually working within the program. To simplify the procedure follow the general outline which I have given in the routine for transferring the variables from the array into the single letter variables.

Line 2005-2010 print a title on the screen so that the data from the experiment can be copied down. The subroutine is printed in full below:-

```
2000 REM **** CALCULATION MODULE ***
2001 REM
2002 REM
2010 PRINT"X=A/B * Y * X=X+B * Y":PRINT
2015 FOR CO=1 TO ND
2020 A=T(1,CO): B=T(2,CO)
2030 J=A/B: K=J*J+Z
2040 L=A+B: M=SQR(ABS(L+Z))
```

```
2110 T(3,CO)=J: T(4,CO)=K
2115 T(5,CO)=L: T(6,CO)=M
2120 PRINTJ,K,L,M
2130 NEXT CO
2135 PRINT"PRESS THE SPACE BAR"
2140 GET A$: IF A$=" " GOTO 2140
2150 RETURN
3000 REM
```

The program is not fully protected against all user entries so please ensure that any special cases which may occur are considered in the calculations ie division by zero and overflow errors.

PET Sort

Nick Marcopoulos, Dataview Ltd.

In issue 7 of the newsletter, a version of Shell-sort in BASIC was presented from W. Murcott. This article introduces a new sorting program which is in fact a modified version of the same sorting algorithm and which has been redesigned from the beginning for PET BASIC. This article also introduces a new methodology for writing more efficient and tailor made routines of similar type for the Commodore PET and explains why a Shellsort type algorithm seems to be more suitable.

A vast amount of time and effort has been already spent on sorting. If you want to get an idea of this just have a glance in the very well known book on this subject written by Donald Knuth (The Art of Computer Programming, Vol 3, Sorting and Searching, Addison Wesley).

To start with there is no 'best' sorting algorithm. All of them become very slow when sorting large arrays. The simplest 'bubble sort' which is in fact very fast for small arrays becomes extremely inefficient for large ones since for a double size array the time needed is four times as much, or, to put it more formally, the sorting speed is proportional to N^2 , where N is the size of the array.

All the sorting algorithms compare the elements of the array and exchange them until the array is sorted. There is no doubt that most of the processing time is spent doing comparisons and exchanges and efforts have been made to minimise the number of the comparisons and exchanges needed as much as possible.

There is a theoretical limit to this minimum so that in the best case the number of comparisons and exchanges is at least proportional to $N \cdot \text{LOG} N$ where LOG is in base 2. Some of the algorithms managed to reach this limit, in fact Quicksort is (in theory) of order $N \cdot \text{LOG} N$ and Shellsort about $1.3 \cdot N \cdot \text{LOG} N$. However all these algorithms have an additional overhead in computation different to comparisons and exchanges. Moreover the actual implementation is done in high level languages and there are substantial

The routine sits in the second cassette buffer and is called by SYS 826. Below are two methods for loading the routine, source code and BASIC loader.

SOUR*.....PAGE 0001

LINE# LOC CODE LINE

```

0001 0000          PTR      = $00
0002 0000          *=$033A
0003 033A A9 00          LDA #$00
0004 033C 85 00          STA PTR      ;LO
0005 033E A9 04          LDA #$04
0006 0340 85 01          STA PTR+1    ;HI BYTE START OF SEARCH
0007 0342 A0 00          LDY #$00     ;ZERO INDIRECT POINTER
0008 0344 20 99 03      ENDL          JSR INCR      ;
0009 0347 B1 00          LDA (PTR),Y   ;EXAMINE LINKS
0010 0349 8D AC 03      STA EP        ;FOR END OF
0011 034C 20 99 03      JSR INCR      ;PROGRAM
0012 034F B1 00          LDA (PTR),Y   ;
0013 0351 0D AC 03      ORA EP        ;
0014 0354 F0 37          BEQ ENDP     ;TOTAL LINK = 0
0015 0356 20 99 03      JSR INCR      ;LINE # LO
0016 0359 20 99 03      JSR INCR      ;LINE # HI
0017 035C A9 00          LDA #$00
0018 035E 8D AD 03      STA QTF      ;CLEAR QUOTE FLAG
0019 0361 8D AE 03      STA REMF     ;CLEAR REM FLAG
0020 0364 20 99 03      NCR          JSR INCR      ;FIRST / NEXT CHAR
0021 0367 B1 00          LDA (PTR),Y   ;
0022 0369 F0 D9          BEQ ENDL     ;END OF BASIC LINE
0023 036B C9 22          CMP #"      ;QUOTES ?
0024 036D F0 1F          BEQ QTMF     ;YES FLIP QUOTE FLAG
0025 036F C9 8F          CMP #$8F     ;IS IT REM
0026 0371 D0 03          BNE CHAR     ;NO
0027 0373 8D AE 03      STA REMF     ;YES SET REM FLAG
0028 0376 CD AF 03      CHAR        CMP SYMBL    ;IS IT SYMBOL
0029 0379 D0 E9          BNE NCR      ;NO NEXT CHARACTER
0030 037B AD AD 03      LDA QTF      ;YES CHECK QUOTE FLAG
0031 037E D0 E4          BNE NCR      ;QUOTE FLAG SET
0032 0380 AD AE 03      LDA REMF
0033 0383 D0 DF          BNE NCR
0034 0385 AD B0 03      LDA TOKEN    ;LOAD KEY TOKEN
0035 0388 91 00          STA (PTR),Y  ;STORE IN BASIC TEXT
0036 038A 4C 64 03      JMP NCR      ;NEXT CHARACTER
0037 038D 60          ENDP          RTS        ;FINISHED TO BASIC
0038 038E AD AD 03      QTMF        LDA QTF
0039 0391 49 01          EOR #$01     ;FLIP QUOTE FLAG
0040 0393 8D AD 03      STA QTF
0041 0396 4C 64 03      JMP NCR
0042 0399 18          INCR          CLC      ;INCREMENT POINTER
0043 039A A5 00          LDA PTR
0044 039C 69 01          ADC #$01
0045 039E 85 00          STA PTR
0046 03A0 A5 01          LDA PTR+1
0047 03A2 69 00          ADC #$00
0048 03A4 85 01          STA PTR+1
0049 03A6 C9 80          CMP #$80
0050 03A8 F0 01          BEQ ERROR
0051 03AA 60          RTS
0052 03AB 00          ERROR        BRK
0053 03AC 00          EP          .BYTE 0
0054 03AD 00          QTF         .BYTE 0
0055 03AE 00          REMF        .BYTE 0
0056 03AF 21          SYMBL       .BYTE '!'
0057 03B0 85          TOKEN       .BYTE $85
0058 03B1          .END

```

ERRORS = 0000


```

10 REM ! TO GIVE INPUT
20 FOR L = 826 TO 944
30 READ C : POKE L , C
40 NEXT L
50 NEW
100 DATA169,0,133,0,169,4,133,1,160,0
110 DATA32,153,3,177,0,141,172,3,32,153
120 DATA3,177,0,13,172,3,240,55,32,153
130 DATA3,32,153,3,169,0,141,173,3,141
140 DATA174,3,32,153,3,177,0,240,217
150 DATA201,34,240,31,201,143,208,3,141
160 DATA174,3,205,175,3,208,233,173,173
170 DATA3,208,228,173,174,3,208,223,173
180 DATA176,3,145,0,76,100,3,96,173,173
190 DATA3,73,1,141,173,3,76,100,3,24
200 DATA165,0,105,1,133,0,165,1,105,0
210 DATA133,1,201,128,240,1,96,0,0,0,0
220 DATA33,133

```

```

10 REM ** DEMONSTRATION / TEST PROGRAM
**
20 SYS 826 : REM ENSURE ALL !'S ARE INP
UT FOR THIS RUN
30 REM THIS IS A TEST !!!
40 ! "WHAT IS YOUR NAME ";N$
50 ! "HOW OLD ARE YOU ";A
60 PRINT N$;" YOU ARE ABOUT";A*365;"DAY
S OLD !!!"
70 GOTO 30

```

User Club & Bits and Pieces

Dave Middleton Software Prizes

In Volume 2 Number 2, Andrew Goltz announced the introduction of the software prize for the best article in CPUCN, the prizes being 50 pounds for the best article in the issue and a 250 pound prize for the best article in the volume. Very little publicity has been given to this and I would suspect that most readers forgot all about it. The decision for who would get the prizes was fairly difficult. Jim Butterfield has been so prolific the he should by rights win all the prizes but that would be a bit pointless as he has access to any software he requires. Also a lot of material is produced 'in house', Mike Gross-Niklaus and Paul Higginbottom are prime examples.

This leaves a fairly small amount of original work produced outside Commodore UK. We have published a lot of small items about programming but there have been very few articles sent in which extend over more than one page but those we have had are generally very good.

Here are the 50 pound software prizes:

No.	Name	Article
1	D.Muir	Digital to Analogue Conversion.
2	Mike Todd	Owners Report.
4	RJ Leman	Assembling an Assembler.
5	LJ Slow	Sorting by Insertion & chaining.
7	AR Clarke	Far infra-red astronomy
	CD Smith	ground station.
7	DA Hills	Supermon Old ROM.
8	D Doyle	DIMP.
8	R Davis	The 'ultimate' screen save.

The 250 pound software prize for volume 2 goes to:

5 Bob Sparks TVA meter for the physics lab.

If the above people would like to write to me giving their choice of software from the current Petpack Master Library catalogue I will arrange to have the programs sent.

I hope this will tempt a few more of you into taking up the pen or better still Wordpro and putting your thoughts, applications, programs or ideas onto paper, it is suprisingly easy once you get started.

Dave Middleton

A review of Adventure

Adventure is a very difficult game to describe because it is so unusual and because if too much information is given away then the game is spoilt.

First a very short history of the game. It was written by two programmers Willie Crowther and Don Woods as the Massachusetts Institute of Technology as an exercise in Artificial Intelligence. The original program was written in FORTRAN which is quite amazing as FORTRAN is not exactly renowned for its string handling capabilities! The game has had an enormous secret following for years as just about every main-frame computer has a copy lurking in the depths of its disk packs which is played by a small band of devotees at enormous cost to the company.

It can hardly be a suprise that PET Adventure has been made available by Jim Butterfield who has converted the FORTRAN code into its BASIC equivalent. The main Adventure program is 12k long and when the text files are included this comes to a massive 56k! To run Adventure you will need a 32k PET and a disk system with Adventure running on drive 0.

This has great significance in making your source listings legible. If you make a nice legible listing, then not only will you be able to understand your own listings when you come to look at them in a years time, but other people should be able to as well.

The following program started as an exercise in managing a strip of memory as a two dimensional array for managing the screen as a window on a sheet (Visicalc do I hear you say). However as per usual, as I began to write this I kept on thinking of other features that would be nice, and so it grew and grew. The listing that follows is quite heavily

commented but never the less a full description of how this works will be necessary next time. If you wish to try the program, there are some systems variables at the start that you may need to change (screen size, sheet width etc.).

This listing uses a lot of symbols whereas the others haven't. The reason for this is that it would have been possible to type in the previous programs into the monitor, or by using SUPERMON or EXTRAMON, but this one is strictly for assembler users only.

More next time...Good luck.

HOPE1.S.....PAGE 0001

LINE#	LOC	CODE	LINE
0001	0000		;*****
0002	0000		;*
0003	0000		;* SYSTEM VARIABLES:-
0004	0000		;* TXTWID - TEXT WIDTH
0005	0000		;* SCRSIZ - SCREEN SIZE
0006	0000		;* (40 OR 80)
0007	0000		;* TEXT IS SELF ORGANISING
0008	0000		;* I.E WIDER TEXT GIVES
0009	0000		;* LESS LINES.
0010	0000		;*
0011	0000		;* BY PAUL HIGGINBOTTOM
0012	0000		;*
0013	0000		;*****
0014	0000		;
0015	0000		TXTWID =132 ;WIDTH OF TEXT
0016	0000		SCRSIZ =80 ;LENGTH OF SCREEN LINE
0017	0000		BLHC =34688 ;BOTTOM LEFT HAND CORNER ADDRESS
0018	0000		;
0019	0000		;ZERO PAGE STORAGE DECLARATIONS
0020	0000		;
0021	0000		LINE =\$23 ;LINE IN TEXT
0022	0000		LOC =\$24 ;POINTER TO SCREEN LOCATION
0023	0000		UNDCSR =\$26 ;CHARACTER UNDER CURSOR
0024	0000		MAXLIN =\$27 ;MAXIMUM LINE NUMBER
0025	0000		TXTEND =\$34 ;TOP OF MEM./END OF TEXT POINTER
0026	0000		RDELAY =\$5E ;DELAY BEFORE REPEAT
0027	0000		SAVCHR =\$5F ;CHARACTER SAVE
0028	0000		REPDY =\$60 ;DELAY BETWEEN REPEATS
0029	0000		IRQSAV =\$61 ;SAVE CURRENT IRQ VECTOR
0030	0000		PTR1 =\$66 ;POINTER TO TEXT LOCATION OF HOME
0031	0000		PTR2 =\$68 ;SPARE POINTER
0032	0000		PTR3 =\$6A ;SPARE POINTER
0033	0000		POSNTX =\$00 ;POSITION ON TEXT LINE
0034	0000		POSNSC =\$01 ;POSITION ON SCREEN LINE
0035	0000		SCRLIN =\$02 ;LINE ON SCREEN
0036	0000		TIMER =\$8F ;JIFFY CLOCK LSD
0037	0000		IRQLO =\$90 ;INTERRUPT VECTOR LO
0038	0000		IRQHI =\$91 ;INTERRUPT VECTOR HI
0039	0000		KEY =\$97 ;UNSHIFTED ASCII KEYPRESS
0040	0000		RVSFLG =\$9F ;REVERSE FLAG
0041	0000		SCREEN =32768 ;SCREEN START ADDRESS
0042	0000		DELAY =\$F4 ;CURSOR BLINK DELAY
0043	0000		;
0044	0000		;ROM ROUTINE DECLARATIONS
0045	0000		;
0046	0000		GET =\$FFE4 ;GET A CHARACTER
0047	0000		PRT =\$FFD2 ;PRINT A CHARACTER
0048	0000		;
0049	0000		*=\$1000 ;STARTS AT 4096 DEC.
0050	1000		;
0051	1000		;SET PTR1 (POINTER 1) TO LO AND HI OF TEXT ADDRESS
0052	1000		;
0053	1000	A9 89	START LDA #<TEXT
0054	1002	85 66	STA PTR1
0055	1004	A9 12	LDA #>TEXT

LINE#	LOC	CODE	LINE
0056	1006	85 67	STA PTR1+1
0057	1008		;
0058	1008		;SET LOC (SCREEN LOCATION) TO HOME POSITION
0059	1008		;
0060	1008	A9 00	LDA #<SCREEN
0061	100A	85 24	STA LOC
0062	100C	A9 80	LDA #>SCREEN
0063	100E	85 25	STA LOC+1
0064	1010		;
0065	1010		;ZEROISE Y AND VARIABLES
0066	1010		;
0067	1010	A0 00	LDY #0
0068	1012	84 00	STY POSNTX ;POSITION ON TEXT
0069	1014	84 01	STY POSNSC ;POSITION ON SCREEN
0070	1016	84 23	STY LINE ;CURRENT LINE IN TEXT
0071	1018	84 02	STY SCRLIN ;CURRENT LINE ON SCREEN
0072	101A	84 9F	STY RVSFLG ;REVERSE FLAG
0073	101C	84 27	STY MAXLIN ;MAXIMUM LINE NUMBER
0074	101E	C6 27	DEC MAXLIN ;SET TO \$FF AT START
0075	1020	A5 90	LDA IRQLO ;SAVE CURRENT IRQ
0076	1022	85 61	STA IRQSAV
0077	1024	A5 91	LDA IRQHI
0078	1026	85 62	STA IRQSAV+1
0079	1028	A2 84	LDX #TXTWID ;SET X TO TEXT WIDTH
0080	102A	78	SEI ;DISABLE INTERRUPTS FOR SPEED
0081	102B		;
0082	102B		;STORE SPACES IN ALL TEXT AREA (32'S)
0083	102B		;
0084	102B	A9 20	CLRTXT LDA #32 ;LOAD A SPACE
0085	102D	91 66	STA (PTR1)Y ;STORE IT
0086	102F	20 18 12	JSR INCPT1 ;INCREMENT PTR1
0087	1032		;
0088	1032		;COUNTS NUMBER OF LINES IN TEXT AREA
0089	1032		;SO THAT IF THE TEXT WIDTH IS SMALL,
0090	1032		;THERE WILL BE A LOT OF LINES AVAILABLE,
0091	1032		;SIMILARLY IF A WIDER TEXT IS USED, THEN
0092	1032		;LESS LINES WILL BE AVAILABLE.
0093	1032		;
0094	1032	CA	DEX ;DECREMENT CHARACTER COUNT
0095	1033	D0 04	BNE CLRT10 ;NOT ZERO - GO ON
0096	1035	A2 84	LDX #TXTWID ;ZERO - SET IT AGAIN
0097	1037	E6 27	INC MAXLIN ;INCREMENT NUMBER OF LINES
0098	1039		;
0099	1039	A5 67	CLRT10 LDA PTR1+1 ;GET HI BYTE OF POINTER
0100	103B	C5 35	CMP TXTEND+1 ;COMPARE WITH END OF MEMORY ?
0101	103D	D0 EC	BNE CLRTXT ;NOT THE SAME - MORE TO DO
0102	103F	A5 66	LDA PTR1 ;THE SAME....
0103	1041	C5 34	CMP TXTEND ;HOW ABOUT THE LO BYTE ?
0104	1043	D0 E6	BNE CLRTXT ;NO - STILL MORE TO DO
0105	1045		;
0106	1045	A9 89	LDA #<TEXT ;SET POINTER TO TEXT START
0107	1047	85 66	STA PTR1
0108	1049	A9 12	LDA #>TEXT
0109	104B	85 67	STA PTR1+1
0110	104D	58	CLI ;RESET INTERRUPTS

LINE#	LOC	CODE	LINE
0111	104E	20 28 12	JSR ONREP ;TURN ON REPEAT
0112	1051		;
0113	1051	78	MAIN SEI ;DISABLE INTERRUPTS FOR SPEED
0114	1052	A9 00	LDA #<SCREEN ;SET PTR2 TO HOME
0115	1054	85 68	STA PTR2
0116	1056	A9 80	LDA #>SCREEN
0117	1058	85 69	STA PTR2+1
0118	105A	A5 66	LDA PTR1 ;TRANSFER PTR1 TO PTR3
0119	105C	85 6A	STA PTR3 ;FOR USAGE
0120	105E	A5 67	LDA PTR1+1
0121	1060	85 6B	STA PTR3+1
0122	1062	A2 19	LDX #25 ;SET SCREEN LINE COUNT
0123	1064		;
0124	1064	A0 4F	DISP10 LDY #SCRSIZ-1 ;SET CHAR. COUNT
0125	1066		;
0126	1066	B1 6A	DISP20 LDA (PTR3)Y ;GET FROM TEXT
0127	1068	91 68	STA (PTR2)Y ;DISPLAY ON SCREEN
0128	106A	88	DEY ;DECREMENT CHARACTER COUNT
0129	106B	10 F9	BPL DISP20 ;>=0 - DO MORE
0130	106D		;
0131	106D	CA	DEX ;DECREMENT LINE COUNT
0132	106E	F0 1C	BEQ DISP99 ;IF ZERO - DONE
0133	1070	18	CLC ;ADD TEXT WIDTH TO PTR1
0134	1071	A5 6A	LDA PTR3
0135	1073	69 84	ADC #TXTWID
0136	1075	85 6A	STA PTR3
0137	1077	A5 6B	LDA PTR3+1
0138	1079	69 00	ADC #0
0139	107B	85 6B	STA PTR3+1
0140	107D		;
0141	107D	18	CLC ;ADD SCREEN SIZE TO PTR2
0142	107E	A5 68	LDA PTR2
0143	1080	69 50	ADC #SCRSIZ
0144	1082	85 68	STA PTR2
0145	1084	A5 69	LDA PTR2+1
0146	1086	69 00	ADC #0
0147	1088	85 69	STA PTR2+1
0148	108A	D0 D8	BNE DISP10 ;AND CARRY ON
0149	108C		;
0150	108C	58	DISP99 CLI ;RE-ENABLE INTERRUPTS
0151	108D		;
0152	108D		;LOC,LOC+1 POINTS TO THE START
0153	108D		;OF THE CURRENT SCREEN LINE.
0154	108D		;
0155	108D		;PROGRAM USES POSNSC (SCREEN POSITION)
0156	108D		;AS Y OFFSET IN LOAD INDIRECT 'LOC'
0157	108D		;TO OBTAIN THE CURRENT CONTENTS
0158	108D		;OF ANY SCREEN LOCATION.
0159	108D		;
0160	108D	A4 01	BEGIN LDY POSNSC ;GET SCREEN POSITION
0161	108F	B1 24	LDA (LOC)Y ;GET CHARACTER
0162	1091	85 26	STA UNDCSR ;SAVE IT FOR LATER
0163	1093	20 1F 12	BEG10 JSR FLIP ;BLINK CURSOR
0164	1096	A9 F4	LDA #DELAY ;SET TIMER TO DELAY
0165	1098	85 8F	STA TIMER ;(JIFFY CLOCK)

LINE#	LOC	CODE	LINE
0166	109A		;
0167	109A		;JSR GET - 'GET'S A CHARACTER FROM
0168	109A		;THE KEYBOARD AND PUTS IT INTO THE
0169	109A		;ACCUMULATOR. A ZERO IS RETURNED,
0170	109A		;IF NO KEY HAS BEEN PRESSED.
0171	109A		;
0172	109A	20 E4 FF	WAIT JSR GET ;GET A CHARACTER
0173	109D	D0 06	BNE GOTONE ;GOT ONE - GO ON
0174	109F	A5 8F	LDA TIMER ;TEST TIMER
0175	10A1	D0 F7	BNE WAIT ;GET AGAIN
0176	10A3	F0 EE	BEQ BEG10 ;FLIP CURSOR AGAIN
0177	10A5		;
0178	10A5	48	GOTONE PHA ;SAVE KEYPRESS ON STACK
0179	10A6	A5 26	LDA UNDCSR ;GET CHAR. UNDER CURSOR
0180	10A8	A4 01	LDY POSNSC ;LOAD OFFSET WITH SCREEN POSITION
0181	10AA	91 24	STA (LOC)Y ;STORE CHAR. BACK THERE
0182	10AC	68	PLA ;GET SAVED KEYPRESS
0183	10AD	C9 1D	CMP #29 ;CURSOR RIGHT ?
0184	10AF	D0 24	BNE GOT5 ;NO - GO ON
0185	10B1		;
0186	10B1		;HANDLE CURSOR RIGHT
0187	10B1		;
0188	10B1	A5 01	RT LDA POSNSC ;GET SCREEN POSITION
0189	10B3	C9 4F	CMP #SCRSIZ-1 ;AT RIGHT HAND SIDE ?
0190	10B5	D0 17	BNE RT10 ;NO - GO ON
0191	10B7	A5 00	LDA POSNTX ;GET TEXT POSITION
0192	10B9	C9 83	CMP #TXTWID-1 ;AT EDGE OF SHEET ?
0193	10BB	D0 03	BNE RT5 ;NO - GO ON
0194	10BD	4C 51 10	JMP MAIN ;YES - CARRY ON
0195	10C0	18	RT5 CLC ;BUMP SCREEN LEFT BY 1
0196	10C1	A5 66	LDA PTR1 ;BY ADDING ONE TO PTR1
0197	10C3	69 01	ADC #1
0198	10C5	85 66	STA PTR1
0199	10C7	A5 67	LDA PTR1+1
0200	10C9	69 00	ADC #0
0201	10CB	85 67	STA PTR1+1
0202	10CD	2C	.BYT \$2C ;SKIP 2 BYTES
0203	10CE		;
0204	10CE	E6 01	RT10 INC POSNSC ;ELSE - BUMP SCREEN POS.
0205	10D0	E6 00	INC POSNTX ;AND TEXT POSITION
0206	10D2	4C 51 10	JMP MAIN ;AND REDISPLAY IT
0207	10D5		;
0208	10D5	C9 12	GOT5 CMP #18 ;REVERSE ?
0209	10D7	D0 07	BNE GOT7 ;NO - GO ON
0210	10D9	A2 80	LDX #\$80 ;YES - SET REVERSE FLAG
0211	10DB	86 9F	STX RVSFLG
0212	10DD	4C 8D 10	JMP BEGIN ;AND CARRY ON
0213	10E0		;
0214	10E0	C9 92	GOT7 CMP #146 ;OFF RVS ?
0215	10E2	D0 07	BNE GOT10 ;NO - GO ON
0216	10E4	A2 00	LDX #\$00 ;YES - RESET REVERSE FLAG
0217	10E6	86 9F	STX RVSFLG
0218	10E8	4C 8D 10	JMP BEGIN ;AND CARRY ON
0219	10EB		;
0220	10EB	C9 9D	GOT10 CMP #157 ;CURSOR LEFT ?

LINE#	LOC	CODE	LINE		
0221	10ED	D0 20		BNE GOT20	;NO - GO ON
0222	10EF				
0223	10EF			;HANDLE CURSOR LEFT	
0224	10EF				
0225	10EF	A5 01	LF	LDA POSNSC	;AT LHS OF SCREEN ?
0226	10F1	D0 15		BNE LF10	;NO - GO ON
0227	10F3	A5 00		LDA POSNTX	;AT EDGE OF SHEET
0228	10F5	D0 03		BNE LF5	;NO - GO ON
0229	10F7	4C 51 10		JMP MAIN	;YES - GO BACK
0230	10FA	38	LF5	SEC	;BUMP SCREEN RIGHT BY 1
0231	10FB	A5 66		LDA PTR1	
0232	10FD	E9 01		SBC #1	
0233	10FF	85 66		STA PTR1	
0234	1101	A5 67		LDA PTR1+1	
0235	1103	E9 00		SBC #0	
0236	1105	85 67		STA PTR1+1	
0237	1107	2C		.BYT \$2C	;SKIP 2 BYTES
0238	1108				
0239	1108	C6 01	LF10	DEC POSNSC	;ELSE BUMP SCREEN POSN
0240	110A	C6 00		DEC POSNTX	;DECREMENT TEXT POSITION
0241	110C	4C 51 10		JMP MAIN	;AND REDISPLAY IT
0242	110F				
0243	110F	C9 11	GOT20	CMP #17	;CURSOR DOWN ?
0244	1111	D0 35		BNE GOT30	;NO - GO ON
0245	1113				
0246	1113	A5 23	DN	LDA LINE	;GET CURRENT LINE
0247	1115	C5 27		CMP MAXLIN	;AT BOTTOM OF TEXT ?
0248	1117	D0 03		BNE DN10	;NO - FINE
0249	1119	4C 8D 10		JMP BEGIN	;YES - NO GO!
0250	111C				
0251	111C	A5 02	DN10	LDA SCRLIN	;ARE WE ON BOTTOM LINE ?
0252	111E	C9 18		CMP #24	
0253	1120	D0 12		BNE DN30	;NO - GO ON
0254	1122				
0255	1122	18	DN20	CLC	;YES - ADD TXTWID TO PTR1
0256	1123	A5 66		LDA PTR1	
0257	1125	69 84		ADC #TXTWID	
0258	1127	85 66		STA PTR1	
0259	1129	A5 67		LDA PTR1+1	
0260	112B	69 00		ADC #0	
0261	112D	85 67		STA PTR1+1	
0262	112F	E6 23		INC LINE	
0263	1131	4C 51 10		JMP MAIN	;AND REDISPLAY IT
0264	1134				
0265	1134	E6 23	DN30	INC LINE	;ELSE BUMP LINE
0266	1136	18		CLC	;AND ADD SCREEN SIZE TO LOC
0267	1137	A5 24		LDA LOC	
0268	1139	69 50		ADC #SCRSIZ	
0269	113B	85 24		STA LOC	
0270	113D	A5 25		LDA LOC+1	
0271	113F	69 00		ADC #0	
0272	1141	85 25		STA LOC+1	
0273	1143	E6 02		INC SCRLIN	
0274	1145	4C 8D 10		JMP BEGIN	;AND CARRY ON
0275	1148				

LINE#	LOC	CODE	LINE
0276	1148	C9 91	GOT30
0277	114A	D0 31	
0278	114C		
0279	114C	A5 02	UP
0280	114E	D0 19	
0281	1150		
0282	1150	A5 23	UP10
0283	1152	D0 03	
0284	1154	4C 8D 10	
0285	1157		
0286	1157	38	UP20
0287	1158	A5 66	
0288	115A	E9 84	
0289	115C	85 66	
0290	115E	A5 67	
0291	1160	E9 00	
0292	1162	85 67	
0293	1164	C6 23	
0294	1166	4C 51 10	
0295	1169		
0296	1169	C6 23	UP30
0297	116B	38	
0298	116C	A5 24	
0299	116E	E9 50	
0300	1170	85 24	
0301	1172	A5 25	
0302	1174	E9 00	
0303	1176	85 25	
0304	1178	C6 02	
0305	117A	4C 8D 10	
0306	117D		
0307	117D	C9 03	GOT40
0308	117F	D0 04	
0309	1181	20 33 12	
0310	1184	60	
0311	1185		
0312	1185	C9 14	GOT43
0313	1187	D0 2B	
0314	1189		
0315	1189	A4 00	DEL
0316	118B	D0 03	
0317	118D	4C 8D 10	
0318	1190		
0319	1190	20 6B 12	DEL5
0320	1193	A4 00	
0321	1195		
0322	1195	B1 68	DEL10
0323	1197	88	
0324	1198	91 68	
0325	119A	C8	
0326	119B	C8	
0327	119C	C0 84	
0328	119E	D0 F5	
0329	11A0		
0330	11A0		

LINE#	LOC	CODE	LINE
0331	11A0		;
0332	11A0	A0 83	LDY #TXTWID-1
0333	11A2	A9 20	LDA #32
0334	11A4	91 68	STA (PTR2)Y
0335	11A6	A5 01	LDA POSNSC ;AT EDGE OF SCREEN ?
0336	11A8	D0 03	BNE DEL20 ;NO - GO ON
0337	11AA	4C EF 10	JMP LF ;YES MOVE LEFT
0338	11AD		;
0339	11AD	C6 01	DEL20 DEC POSNSC ;ELSE DECR. SCRIN POS
0340	11AF	C6 00	DEC POSNTX ;AND TEXT POSITION
0341	11B1	4C 51 10	JMP MAIN ;AND REDISPLAY IT
0342	11B4		;
0343	11B4	C9 94	GOT46 CMP #148 ;INSERT ?
0344	11B6	D0 1E	BNE GOT50 ;NO - GO ON
0345	11B8		;
0346	11B8	20 6B 12	INS JSR CALC ;CALCULATE POSITION
0347	11BB		;
0348	11BB	A0 82	INS5 LDY #TXTWID-2 ;START AT RHS OF TEXT
0349	11BD		;
0350	11BD	C4 00	INS10 CPY POSNTX ;DONE ?
0351	11BF	90 0C	BCC INS20 ;YES - GO ON
0352	11C1	B1 68	LDA (PTR2)Y ;GET A CHARACTER
0353	11C3	C8	INY ;MOVE ONTO NEXT SQUARE
0354	11C4	91 68	STA (PTR2)Y ;PUT IT THERE
0355	11C6	88	DEY ;BACK TO WHERE WE WERE
0356	11C7	F0 04	BEQ INS20 ;IF IN 1ST - STOP
0357	11C9	88	DEY ;AND ONTO THE NEXT
0358	11CA	4C BD 11	JMP INS10 ;AND CARRY ON
0359	11CD		;
0360	11CD	A4 00	INS20 LDY POSNTX ;GET OFFSET
0361	11CF	A9 20	LDA #32 ;AND PUT SPACE THERE
0362	11D1	91 68	STA (PTR2)Y
0363	11D3	4C 51 10	JMP MAIN ;CARRY ON
0364	11D6		;
0365	11D6	C9 13	GOT50 CMP #19 ;HOME ?
0366	11D8	D0 1D	BNE GOT60 ;NO - GO ON
0367	11DA		;
0368	11DA	A9 89	HM LDA #<TEXT ;RESET POINTER
0369	11DC	85 66	STA PTR1
0370	11DE	A9 12	LDA #>TEXT
0371	11E0	85 67	STA PTR1+1
0372	11E2	A9 00	LDA #<SCREEN ;AND LOC
0373	11E4	85 24	STA LOC
0374	11E6	A9 80	LDA #>SCREEN
0375	11E8	85 25	STA LOC+1
0376	11EA	A0 00	LDY #0 ;AND VARIABLES
0377	11EC	84 00	STY POSNTX ;TEXT POSITION
0378	11EE	84 01	STY POSNSC ;SCREEN POSITION
0379	11F0	84 23	STY LINE ;AND LINE
0380	11F2	84 02	STY SCRLIN ;AND SCREEN LINE
0381	11F4	4C 51 10	JMP MAIN ;AND CARRY ON
0382	11F7		;
0383	11F7	C9 93	GOT60 CMP #147 ;CLR HME ?
0384	11F9	D0 06	BNE OUT10 ;NO - GO ON
0385	11FB	20 33 12	JSR OFFREP ;YES - DISABLE REPEAT

LINE#	LOC	CODE	LINE
0386	11FE	4C 00 10	JMP START ;AND START AGAIN
0387	1201		;
0388	1201		;CONVERT KEYPRESS TO SCREEN CODE
0389	1201		;
0390	1201	48	OUT10 PHA ;SAVE IT
0391	1202	0A	ASL A ;PUT BIT 7 INTO CARRY
0392	1203	68	PLA ;REGET IT
0393	1204	29 3F	AND #\$3F ;MASK OUT TOP 2 BITS
0394	1206	90 02	BCC OUT20 ;IF BIT7 SET - GO ON
0395	1208	09 40	ORA #\$40 ;ELSE SET BIT6
0396	120A		;
0397	120A	05 9F	OUT20 ORA RVSFLG ;NEGATE IT IF IN RVS
0398	120C	48	PHA ;SAVE IT FOR LATER
0399	120D	20 6B 12	JSR CALC
0400	1210	A4 00	LDY POSNTX ;SET OFFSET
0401	1212	68	PLA ;REGET CHARACTER
0402	1213	91 68	STA (PTR2)Y ;STORE IT
0403	1215	4C B1 10	JMP RT ;AND DO A CURSOR RIGHT
0404	1218		;
0405	1218		;SUBROUTINE TO DO A 16 BIT INCREMENT ON PTR1
0406	1218		;
0407	1218	E6 66	INCPT1 INC PTR1 ;INCREMENT LO BYTE
0408	121A	D0 02	BNE NOINCL ;NOT ZERO - GO ON
0409	121C	E6 67	INC PTR1+1 ;ELSE - INCREMENT HI BYTE
0410	121E	60	NOINCL RTS ;AND RETURN
0411	121F		;
0412	121F		;SUBROUTINE TO NEGATE SCREEN POSITION
0413	121F		;
0414	121F	A4 01	FLIP LDY POSNSC ;LOAD OFFSET
0415	1221	B1 24	LDA (LOC)Y ;GET IT
0416	1223	49 80	EOR #\$80 ;FLIP IT
0417	1225	91 24	STA (LOC)Y ;STORE IT BACK
0418	1227	60	RTS ;AND RETURN
0419	1228		;
0420	1228		!-----TXTWID-----!
0421	1228		!-----!
0422	1228		TEXT-----SCRSIZ-----!
0423	1228		!-----!
0424	1228		!-----!
0425	1228		!-----!
0426	1228		!-----!
0427	1228		!-----!
0428	1228		!-----!
0429	1228		!-----!
0430	1228		!-----!
0431	1228		!-----!
0432	1228		!-----!
0433	1228		!-----!
0434	1228		!-----!
0435	1228		!-----!
0436	1228		!-----!
0437	1228		!-----!
0438	1228	78	ONREP SEI ;DISABLE INTERRUPTS
0439	1229	A9 3E	LDA #<REPEAT ;CHANGE IRQ TO REPEAT
0440	122B	85 90	STA IRQLO

LINE#	LOC	CODE	LINE
0441	122D	A9 12	LDA #>REPEAT
0442	122F	85 91	STA IRQHI
0443	1231	58	CLI ;RE-ENABLE INTERRUPTS
0444	1232	60	RTS ;AND RETURN
0445	1233		;
0446	1233		;ROUTINE TO DISABLE REPEAT FUNCTION
0447	1233		;BY CHANGING THE INTERRUPT VECTOR
0448	1233		;TO ITS ORIGINAL LOCATION.
0449	1233		;
0450	1233	78	OFFREP SEI ;DISABLE INTERRUPTS
0451	1234	A5 61	LDA IRQSAV ;RESET SAVED IRQ
0452	1236	85 90	STA IRQLO
0453	1238	A5 62	LDA IRQSAV+1
0454	123A	85 91	STA IRQHI
0455	123C	58	CLI ;RE-ENABLE INTERRUPTS
0456	123D	60	RTS
0457	123E		;
0458	123E	A5 97	REPEAT LDA KEY ;GET LAST KEY
0459	1240	C5 5F	CMP SAVCHR ;SAME AS LAST TIME ?
0460	1242	F0 0D	BEQ REP100 ;YES
0461	1244	85 5F	STA SAVCHR ;NO - SAVE NEW CHAR.
0462	1246	A9 10	LDA #\$10 ;INITIALISE DELAY
0463	1248	85 5E	STA RDELAY
0464	124A	A9 04	LDA #\$04 ;INITIALISE REPEAT DELAY
0465	124C	85 60	STA REPDY
0466	124E		;
0467	124E	6C 61 00	REPEXT JMP (IRQSAV)
0468	1251		;
0469	1251	C9 FF	REP100 CMP #\$FF ;NO KEY ?
0470	1253	F0 F9	BEQ REPEXT ;YES - NO MORE
0471	1255	A5 5E	LDA RDELAY ;TIME TO REPEAT ?
0472	1257	F0 04	BEQ REP200 ;YES - DO IT
0473	1259	C6 5E	DEC RDELAY ;ELSE DECREMENT DELAY
0474	125B	D0 F1	BNE REPEXT ;AND EXIT
0475	125D		;
0476	125D	C6 60	REP200 DEC REPDY ;BETWEEN REPEAT READY ?
0477	125F	D0 ED	BNE REPEXT ;NO - EXIT
0478	1261	A9 03	LDA #\$03 ;YES - RESET DELAY
0479	1263	85 60	STA REPDY
0480	1265	A9 00	LDA #\$00 ;SET NO KEY
0481	1267	85 97	STA KEY ;FOR REPEAT
0482	1269	F0 E3	BEQ REPEXT ;AND EXIT
0483	126B		;
0484	126B		;WORK OUT ADDRESS OF START OF TEXT
0485	126B		;LINE INTO PTR2,PTR2+1
0486	126B		; = TEXT + LINE * TXTWID
0487	126B		;
0488	126B	A9 89	CALC LDA #<TEXT ;SET PTR2 TO TEXT ADDRESS
0489	126D	85 68	STA PTR2
0490	126F	A9 12	LDA #>TEXT
0491	1271	85 69	STA PTR2+1
0492	1273	A6 23	LDX LINE ;SET COUNTER TO LINE
0493	1275	4C 86 12	JMP CALC20 ;AND SEE IF DONE
0494	1278		;
0495	1278	18	CALC10 CLC ;ADD TEXTWIDTH TO RESULT

HOPE1.S.....PAGE 0010

LINE#	LOC	CODE	LINE
0496	1279	A5 68	LDA PTR2
0497	127B	69 84	ADC #TXTWID
0498	127D	85 68	STA PTR2
0499	127F	A5 69	LDA PTR2+1
0500	1281	69 00	ADC #0
0501	1283	85 69	STA PTR2+1
0502	1285	CA	DEX ;DECREMENT COUNTER
0503	1286	D0 F0	CALC20 BNE CALC10 ;IF NOT ZERO - DO MORE
0504	1288		;
0505	1288	60	CALC30 RTS ;RETURN
0506	1289		;
0507	1289		TEXT =* ;TEXT STARTS HERE !
0508	1289		;
0509	1289		.END

ERRORS = 0000

SYMBOL TABLE

SYMBOL	VALUE
BEG10	1093
CALC10	1278
CLRTXT	102B
DEL5	1190
DISP99	108C
DN30	1134
GOT20	110F
GOT46	11B4
GOT7	10E0
INS	11B8
IRQHI	0091
LF	10EF
LOC	0024
OFFREP	1233
POSNSC	0001
PTR2	0068
REP200	125D
RT	10B1
SAVCHR	005F
START	1000
TXTWID	0084
UP20	1157
BEGIN	108D
CALC20	1286
DEL	1189
DELAY	00F4
DN	1113
FLIP	121F
GOT30	1148
GOT5	10D5
GOTONE	10A5
INS10	11BD
IRQLO	0090
LF10	1108
MAIN	1051
ONREP	1228
POSNTX	0000
PTR3	006A
RPDY	0060
RT10	10CE
SCREEN	8000
TEXT	1289
UNDCSR	0026
UP30	1169
BLHC	8780
CALC30	1288
DEL10	1195
DISP10	1064
DN10	111C
GET	FFE4
GOT40	117D
GOT50	11D6
HM	11DA
INS20	11CD
IRQSAV	0061
LF5	10FA
MAXLIN	0027
OUT10	1201
PRT	FFD2
RDELAY	005E
REPEAT	123E
RT5	10C0
SCRLIN	0002
TIMER	008F
UP	114C
WAIT	109A
CALC	126B
CLRT10	1039
DEL20	11AD
DISP20	1066
DN20	1122
GOT10	10EB
GOT43	1185
GOT60	11F7
INCPT1	1218
INS5	11BB
KEY	0097
LINE	0023
NOINCL	121E
OUT20	120A
PTR1	0066
REP100	1251
REPEXT	124E
RVSFLG	009F
SCR512	0050
TXTEND	0034
UP10	1150

END OF ASSEMBLY

Interrupt on the Commodore PET

Brad Templeton

One of the most important features of the COMMODORE PET operating system is the use of interrupts. They are used to reset the PET, and they handle most of the tape and all of the keyboard i/o. This article will provide an introduction to interrupts on the 6502 (The PET's cpu) and a description of how the PET handles them. For your information, pseudo source listing is provided for the interrupt software of the PET, as produced by my disassembler.

Under normal conditions a processor executes machine code in a linear fashion. It moves through memory, obtaining instructions (which can be one, two or three bytes long) and executing them. Sometimes, certain programmed instructions cause jumps to other places, just like GOTO and GOSUB of BASIC. To make a machine more flexible, however, interrupts are provided to do jobs that would be very expensive to do in Software.

Essentially, an interrupt is controlled by a line right into the processor. When the processor detects the correct voltage on this line, an interrupt may be generated. First, in order to simplify matters, the processor finishes the instruction it is presently carrying out. Then, if the interrupt is ok (interrupts can be masked), the processor saves the program location it was at, and the contents of its flags onto the stack. It then goes to a special reserved area of memory (in ROM on the PET) and pulls out two bytes indicating what location it should start executing from. It then goes there and executes machine code until the instruction RTI (Return from Interrupt \$40) is encountered. It then goes back to the stack and restores its flags, and loads the location it saved to the instruction counter. It then goes and executes the code after where it stopped as though nothing has occurred. (If the interrupt program was correctly written).

On the 6502, three types of hardware interrupts can occur, as well as a fourth special type. The locations they branch to are kept in byte pairs called vectors at the end of memory. One of these interrupts, NMI or Non Maskable Interrupt, cannot be used on the PET. Its Vector, \$FFFA-B, points to \$CA60, which is the middle of a subroutine. The line for this is also fixed off by a resistor on the PC board. Later PETs may plan to include this.

The interrupt called for power up is named RES. It branches to a routine which sets up basic and the operating system. It also, through what I consider to be one of the PET's worst design

flaws, branches to the routine to destructively test how much memory is in the machine. At the very start, it also tests the condition of the diagnostic sense (MSB of \$E810) and goes to the diagnostic routine if this is set. RFS is fired by power up, or by grounding pin 27 on the bottom of your memory expansion bus. If you set it by touching that pin, it does not clear memory below \$400. So programs there (the tape buffers) are safe. This is, unfortunately, a very small area. It vectors through \$FFFC-D.

The general use, hardware interrupts is the IRQ. IRQ vectors through \$FFFE-F, as does BRK. This points to location \$E66B in the PET. It is generated every 60th of a second by the tv hardware, on pin 28. It is also connected to the 6522 versatile interface adaptor. I will discuss the 60 per second interrupts here in detail. For information of generation by the 6522 (there is another whole article's worth of material in there) you can see "The User Port Cookbook" (1). Interrupts can be generated from it at exactly timed intervals, and by certain i/o conditions on the user port and IEEE bus. The exactly timed intervals are used to send precise frequency signals to the tape. (In fact, the 6522 is the PET's tape interface!)

The 60 per second interrupts do the following:

Scan the keyboard, checking for new keys and decoding them.
Increment the real time clock, and check for midnight.
Flash the cursor if it is on (\$0224 = 0).
Test tape recorded status for stop-start.
Copy a byte for the break key test.
Whatever else you want them to do.

When the IRQ occurs, the code at \$E668 (see source) saves the processor register A, X and Y on the stack. It then checks, by loading back from the stack, the flags, to see if the BRK flag was set. The BRK, a software IRQ, vectors through the same place, but sets the BRK flag. This is handy to test what type of interrupt occurred. It then does a jump indirect to one of two places in RAM (\$219 or \$218) depending on the type of interrupt.

Normally, the RAM IRQ vector is set to \$F685, which is the standard IRQ code. BRK has no default setting. The small piece of code you see after the JMP indirects is the return code, which restores the registers and does the RTI. The first thing INT-CODE does is the JSR INCR-CLOCK which increments the clock and copies the PIA register which the STOP key test uses. When Steve Punter of Mississauga saw this with the disassembler, he devised an ingenious way to disable the BREAK key of the PET. By telling the PET to branch to \$E688 instead of \$E685 by means of a POKE 537, 136 statement, the PET bypasses the INCR-CLOCK subroutine, and does not test

the break key. (Note INCR-CLOCK passes through a JMP vector table in high ROM at \$FEEA). This has the side effect of turning off the real time clock. When this statement is not used the clock proceeds normally. After it is updated, it is compared with a three byte table that contains the value for midnight. If it is midnight on the clock, it is zeroed. The PET also keeps a secondary clock just after the main one. This is used for calibrating the real time clock. About every 6 seconds, this clock reaches a special limit, and when it does, it is zeroed, and the main clock is not incremented on this cycle. This is because the interrupt generator runs slightly faster than exactly 60 times per second. Even with this compensation, you may have noticed the clock is a few seconds off after several hours of PET operation. If they had used the 60 hz ac power line for the interrupt, it would have been more accurate, but that would have caused problems for PETs sold abroad.

After doing the clock, it proceeds to flash the cursor, once every third of a second, if the location FLASHING (\$224) is set to zero. (POKE 548,0 in a program turns the cursor on, but with some bugs - try it and see). It does it with a very silly method that has no purpose, instead of the standard method, a \$80. It then sets up two keyboard test locations.

In using your PET, you may have noticed that if the tape drive is stopped by the machine itself, that you can push stop and play and the motor will run again. After this comes the keyboard interpretation routines. The method of decoding the keyboard PIA has already been published in your PET manual, and in PET user notes so I will not dwell on it here. Once it has the matrix co-ordinate of the key, it waits for it to stabilize, to avoid bounce and repeating letters. (The TRS-80 does this poorly). It then converts the matrix number to an ascii character through the table at \$E75C. (You can see this table in your programs. If you want to account for how long a key is held down - a great real time feature!) It then puts the key in the correct place in the keyboard buffer starting at \$20F. Finally it goes back.

What Can You Do?

Because the PET IRQ goes through RAM, it is one of the main links you have that can give you operating system control. You can insert your own programs before and after the interrupt code to have your PET do two jobs at once, like handle i/o while running BASIC. I have used interrupts to write programs to:

Interrupt the PET keyboard and the full sized keyboard I attached to the PET like a regular keyboard. Provide functions like repeat after a certain period of time and shift lock.

Turn the ! key to a statement number key, so that it would provide a line number 10 higher with every push. Have upper case letter keys print out as full BASIC keywords. Display whole pages of PET memory constantly on the screen. Provide a non-destructive reset that works in special cases. Much more is possible.

To use your own programs, you merely set them up in some convenient location (machine code only), preferably starting at location that ends in \$85 (BASIC2: \$2E), such as \$0385 in the second tape buffer (Uses 1st cassette buffer in BASIC2: \$032E). Something located there can then be started with a POKE 538,3 and stopped with POKE 538,230 (BASIC2: Start with POKE145,3 and stop with POKE145,230), rather than having to write a special machine language program that disables the interrupt with SEI, changes the locations and enables the interrupt with CLI. You do not need to disable if you are only changing one byte of the location. Put some code there and follow it with a JMP \$E685 (BASIC2: \$E62E). This way it does your code and proceeds on to do its own. If you put in the following series:-

```
BASIC1:
  0385  FE 50 80 4C 85 E6
```

```
BASIC2:
  032E  FE 50 80 4C 2E E6
```

and initiate it with POKE 538,3 (BASIC2: POKE 145,3), you will see a byte on the screen constantly increasing in 'value' once every 60th of a second. The PET will also be doing everything else as usual. The following code:

```
BASIC1:
  0385  A2 00 BD 00 00 9D 50 80
  038D  E8 D0 F7 4C 85 E6 00 00
```

```
BASIC2:
  032E  A2 00 BD 00 00 9D 50 80
  0336  E8 D0 F7 4C 2E E6 00 00
```

will dump a page of memory on the screen constantly. You can POKE 905 (BASIC2: POKE 818) with the page you wish to examine. Try 0,1,2,3,4,31,232. It starts with page 0. When scanning page 0 move the cursor and see what happens.

While doing this, you may have noticed that there is no flicker whatsoever on the screen despite the massive amount of writing to it being done. (Far faster than BASIC printing). This is because the interrupt is fired by the screen scan signal and the screen is doing nothing shortly after the interrupt goes. This is also why the flashing cursor will never "snow" the screen. You can store almost half a screen without "snow" this way.

Sometimes it is important to put code in after the interrupt code of the PET.

This can be done by manipulation of the stack, and is necessary for programs like the statement numberer or keyboard I included in my list above.

It should be noted that probably the only reason the IRQ vector is in RAM is that the PET does change it for tape I/O routines. There is a table of possible vectors starting at \$FD28 in the ROM, and the table ends with the standard vector \$E685. If you ever change the high order byte of the IRQ RAM vector, you must reset it before tape I/O is done. If you don't, the PET will reset it anyway, but the tape I/O may not be done, and you may crash your PET.

Notes for BASIC2 PETs

H.A.E. Broomhall

These notes are intended to be read with the article by Brad Templeton. They detail the differences of the BASIC2 PET from the BASIC1.

NMI

This can now be used. The vector points to \$FCFE. At this address is a single instruction JMP (\$0094). This is a 'JUMP indirect' instruction, and uses as the target address a 16 bit address located in \$94 (low byte) and \$95 (high byte). On power up this is set to \$C389, which is the entry to BASIC in command mode, and gives the familiar 'READY' prompt and the cursor.

The importance of NMI is due to the high priority given to it by the 6502. The micro will react to NMI even during an ordinary interrupt (called by IRQ), and also during a 'hangup' in a machine code routine. In the mode to which it is set by power-up,

the return to BASIC jump will not always clear a 'hang', as other memory locations may not be as the PET expects! NMI is most useful when the PET is used to control external devices in a multiple interrupt situation.

RES

The only difference here is that the diagnostic routines have been removed, and the diagnostic jumper causes an entry to the machine language monitor, which is resident in the MkII PETs. This fact is the basis of a 'decrash' device available from some outlets.

IRQ

The start of the IRQ routine in MkII PETs is at \$E61B. The routine is broadly similar to that in MkI PETs. The main differences in layout are:-

The exit routine 'RETURN-INT' occurs at the end of the 'INT-CODE' section (addr \$E6E4).

The subroutine 'DECODE-KBD' is incorporated into 'INT-CODE'.

The main differences in detail:-

IRQ vector address:- \$90,\$91

BRK vector address:- \$92,\$93

IRQ vector normally points to \$E62E

To stop clock and 'BREAK' test:- POKE144,49
To reset to normal:- POKE144,46

Cursor flash flag (was \$0224) is at \$A7. The flashing cursor routine is no longer silly (someone must have read the article!); EOR #\$80 is used.

The keyboard matrix to ASCII conversion table is now at \$E6F8. The keyboard buffer now at \$026F.

PRICE BREAKTHROUGH

PROGRAMMER'S
TOOLKIT

£29

TOOLKIT WITH
EXTENSION BOARD

£42

PETMASTER
SUPERCHIP

£45

Add 15% VAT to all prices



SUPERSOFT

28 Burwood Avenue — Eastcote
Pinner — Middlesex
Phone: 01 866 3326 anytime



small systems engineering limited

2-4 Canfield Place London NW6 3BT Telephone 01-328 7145 6

IEEE-488 PET INTERFACES

B200 Bi-directional RS232C serial £186.00

Type C Uni-directional RS232C serial £120.00

AP Addressable parallel for Centronics or Anadex printers £106.00

GPI AP Micro based bi-directional serial interface with buffering
Custom GPI software development for special interfacing requirements £249.00

All serial interfaces incorporate:

- Software or switched Baud rate selection with 16 different rates selectable
- Crystal controlled Baud rate
- Full RS232C handshake
- 20 mA current loop i.o. capability

All the above interfaces have two modes of code conversion to match print out to the PET screen for either display mode

Non Addressable parallel £45.00
TV/Video interface £35.00

We also stock a range of PET connectors.

PET SOFTWARE

TCL PASCAL BASIC COMPILER COMMUNICATOR - Intelligent terminal package teletype, DEC VT50 emulation, runs on 80 column PETs
VISICALC WORDCRAFT - Word processing software for 40 and 80 column PETs

NEW... MUPET

Multi user disk system allows 2 to 8 PETs to be linked together to share a CBM disk unit and printer: prices from £595 for a 3 user system.

RICOH R.P. 1600 Daisy Wheel Printer

- Printing speed - 60 C.P.S.
 - 124 character print wheel
 - Integral PET Interface
- Complete word processing systems - PET, Wordcraft, R.P.1600 also available**
Price R.P.1600 £1590 Inc PET Interface

ANADEx D.P. 9500/01 Line Printer

- Bi-directional printing with shortest distance sensing logic
- High density graphics
- 50 to 200 +
- Parallel, RS232C and Current Loop interfaces standard. PET Interfaces available.

Prices: 9500 - £895 9501 - £995

Full range of PET computers and peripherals

We can offer expert advice on scientific and industrial applications.

PET / 6502 PROGRAMMERS

If you live in the Birmingham area and wish to join a foremost Software House specialising in programs for the PET then we would be delighted to hear from you. A sound working knowledge of the PET and the CBM disk is essential for which the rewards in terms of salary, bonuses, job satisfaction and a friendly atmosphere will be substantial.

Please write with full details of your career and experience to date to our recruitment consultants:

**225 Graphics Ltd
Security Buildings
536 Hobmoor Road
Yardley
Birmingham
B25 8TN**

Harry Broomhall — Disk Wizard

Volume 2 Issue 8 was brought out with a great deal of help from Harry Broomhall who has a program called LAZARUS, which can bring a disk which is apparently dead, back to life. The various Commodore centres around the world each produce their own Newletters and information is passed freely amongst the Editors. In Issue 2.8 many of the articles came from the Canadian newsletter 'The Transactor'. Karl Hildon the Editor, sent me the Transactor on disk but the disk had been corrupted by allowing DOS2.1 to write to a DOS1 disk which is a no-no! Harry put the disk right by feeding it to Lazarus. The program can not work miracles however, some disks are just too far gone! If you have a disk which contains valuable data and it has been corrupted in some manner, Harry can put it right for you. The cost for this is £150 if he can get the disk working again. Contact Harry Broomhall at Heronview Ltd., 3 Errol St., London EC1.

BASIC2/BASIC4 ROM Comparison

BASIC2.0 and BASIC4.0 ROM Memory Maps

Entry points comparison for BASIC 2.0 and BASIC 4.0 based on Memory maps supplied by Jim Butterfield, compilation into 1 table by Dave Briggs.

This map shows where various routines lie. The first address is not necessarily the proper entry point for the routine. Similarly, many routines require register setup or data preparation before calling.

BASIC 2.0	BASIC 4.0	Description
C000-C045	B000-B065	Action addresses for primary keywords
C046-C073	B066-B093	Action addresses for functions
C074-C091	B094-B0B1	Hierarchy and action addresses for operators
C092-C192	B0B2-B20C	Table of Basic keywords
C193-C2A9	B20D-B321	Basic messages, mostly error messages
C2AA-C2D7	B322-B34F	Search the stack for FOR or GOSUB activity
C2D8-C31A	B350-B392	Open up space in memory
C31B-C327	B393-B39F	Test: stack too deep?
C328-C354	B3A0-B3CC	Check available memory
C355	B3CD	Send canned error message, then:
C389-C3AA	B3FF-B41E	Warm start; wait for Basic command
C3AB-C441	B41F-B4B5	Handle new Basic line input
C442-C46E	B4B6-B4E1	Rebuild chaining of Basic lines
C46F-C494	B4E2-B4FA	Receive line from keyboard
C495-C52B	B4FB-B5A2	Crunch keywords into Basic tokens
C52C-C55A	B5A3-B5D1	Search Basic for given line number
C55B	B5D2	Perform NEW, and;
C577-C5A6	B5EC-B621	Perform CLR
C5A7-C5BA	B622-B62F	Reset Basic execution to start
C5B5-C657	B630-B6DD	Perform LIST
C658-C6FF	B6DE-B784	Perform FOR
C700-C72F	B785-B7B6	Execute Basic statement
C730-C73E	B7B7-B7C5	Perform RESTORE
C73F-C76A	B7C6-B7ED	Perform STOP or END
C76B-C784	B7EE-B807	Perform CONT
C785-C78F	B808-B812	Perform RUN
C790-C7AC	B813-B839	Perform GOSUB
C7AD-C7D9	B830-B85C	Perform GOTO
C7DA	B85D	Perform RETURN, then:
C7F3-C80D	B883-B890	Perform DATA: skip statement
C80E-C810	B891-B893	Scan for next Basic statement
C811-C82F	B894-B8B2	Scan for next Basic line
C830	B8B3	Perform IF, and perhaps:
C843-C852	B8C6-B8D5	Perform REM: skip line
C853-C872	B8D6-B8F5	Perform ON
C873-C8AC	B8F6-B92F	Accept fixed-point number
C8AD-C927	B930-BA87	Perform LET
C98B-C990	BA88-BA8D	Perform PRINT#
C991-C9A4	BA8E-BAAL	Perform CMD
C9A5-CA1B	BAA2-BB1C	Perform PRINT
CA1C-CA38	BB1D-BB39	Print string from memory
CA39-CA4E	BB3A-BB4B	Print single format character
CA4F-CA7C	BB4C-BB79	Handle bad input data
CA7D-CAAA	BB7A-BBA3	Perform GET
CAA7-CAC0	BBA4-BBBD	Perform INPUT#
CAC1-CAF9	BBBE-BBF4	Perform INPUT
CAFA-CB06	BBF5-BC01	Prompt and receive input
CB07-CBFB	BC02-BCF6	Perform READ
CBFC-CC1F	BCF7-BD18	Canned Input error messages
CC20-CC78	BD19-BD71	Perform NEXT
CC79-CC9E	BD72-BD97	Check type mismatch
CC9F-CDEB	BD98-BEE8	Evaluate expression
CDEC-CDF1	BEE9-BEEF	Evaluate expression within parentheses
CDF2-CE02	BEEF-EEEF	Check parenthesis, comma
CE03-CE07	BF00-BF0B	Syntax error exit
CE0F-CE88	BF8C-C046	Variable name setup
CE88-CE0E	C047-C085	Set up function references
CEC8-CEF7	C086-C0B5	Perform OR, AND
CEF8-CF5F	C0B6-CL1D	Perform comparisons
CF60-CF6C	CL1E-CL2A	Perform DIM
CF6D-CFF6	CL2B-CLBF	Search for variable
D001-D077	CLC0-C2C7	Create new variable
D078-D088	C2C8-C2D8	Setup array pointer
D089-D08C	C2D9-C2DC	32768 in floating binary
D08D-D0AB	C2DD-C2FB	Evaluate integer expression
D0AC-D227	C2FC-C447	Find or make array
D228-D258	C477-CA47	Compute array subscript size
D259	C4A8	Perform FRE, and:
D26D-D279	C4BC-C4C8	Convert fixed-to-floating
D27A-D27F	C4C9-C4CE	Perform POS
D280-D28C	C4CF-C4DB	Check not Direct
D28D-D2BA	C4DC-C509	Perform DEF
D2BB-D2CD	C50A-C51C	Check FNx syntax
D2CE-D33E	C51D-C58D	Evaluate FNx
D33F-D34E	C58E-C59D	Perform STR\$
D34F-D360	C59E-C5AF	Do string vector
D361-D3CD	C5B0-C61C	Scan, set up string
D3CE-D3FF	C61D-C669	Allocate space for string
D400-D496	C66A-C74E	Garbage collection
D517-D553	C74F-C78B	Concatenate
D554-D57C	C78C-C7B4	Store string
D57D-D5B4	C7B5-C810	Discard unwanted string
D5B5-D5C5	C811-C821	Clean descriptor stack
D5C6-D5D9	C822-C835	Perform CHR\$
D5DA-D605	C836-C861	Perform LEFT\$
D606-D610	C862-C86C	Perform RIGHT\$
D611-D63A	C86D-C896	Perform MID\$
D63B-D655	C897-C8B1	Pull string data
D656-D65B	C8B2-C8B7	Perform LEN
D65C-D664	C8B8-C8C0	Switch string to numeric
D665-D674	C8C1-C8D0	Perform ASC
D675-D686	C8D1-C8E2	Get byte parameter
D687-D6C5	C8E3-C920	Perform VAL
D6C6-D6D1	C921-C92C	Get two parameters for POKE or WAIT
D6D2-D6E7	C92D-C942	Convert floating-to-fixed
D6E8-D706	C943-C959	Perform PEEK
D707-D70F	C95A-C962	Perform POKE
D710-D72B	C963-C97E	Perform WAIT
D72C-D732	C97F-C985	Add 0.5 to accumulator #1
D733-D744	C986-C997	Perform subtraction
D76E-D852	C998-CA7C	Perform addition
D853-D889	CA7D-CAB3	Complement accum#1
D88A-D88E	CAB4-CAB8	Overflow exit
D88F-D8C7	CAB9-CAF1	Multiply-a-byte
D8C8-D8F5	CAF2-CB1F	Constants
D8F6-D936	CB20-CB5D	Perform LOG
D937-D964	CB5E-CBC1	Perform multiplication
D968-D9C2	CBCE-CBEC	Unpack memory into accum#2
D9C3-D9DF	CBED-CC09	Test & adjust accumulators
D9E0-D9ED	CC0A-CC17	Handle overflow and underflow
D9EE-DA04	CC18-CC2E	Multiply by 10
DA05-DA09	CC2F-CC33	10 in floating binary
DA0A-DA12	CC34-CC3C	Divide by 10
DA1E-DAAD	CC3D-CC44	Perform divide-by
DA13-DALD	CC45-CCD7	Perform divide-into
DAAE-DAD2	CCD8-CCFC	Unpack memory into accum#1
DAD3-DB07	CCFD-CD31	Pack accum#1 into memory
DB08-DB17	CD32-CD41	Move accum#2 to #1
DB18-DB26	CD42-CD50	Move accum#1 to #2
DB27-DB36	CD51-CD60	Round accum#1
DB37-DB44	CD61-CD6E	Get accum#1 sign
DB45-DB63	CD6F-CD8D	Perform SGN
DB64-DB66	CD8E-CD90	Perform ABS
DB67-DBA6	CD91-CDD0	Compare accum#1 to memory
DBA7-DBD7	CDDE-CE01	Floating-to-fixed
DBD8-DBFE	CE02-CE28	Perform INT
DBFF-DC89	CE29-CEB3	Convert string to floating-point
DC8A-DCBE	CEB4-CEB8	Get new ASCII digit
DCBF-DCCD	CEE9-CEFB	Constants
DCEC	CF78	Print IN, then:
DCD9-DCES	CF7F-CF92	Print Basic line #
DCE9-DE1C	CF93-D0C6	Convert floating-point to ASCII
DE1D-DE5D	D0C7-D107	Constants
DE5E-DE67	D108-D111	Perform SQR
DE68-DEA0	D112-D14A	Perform power function
DEA1-DEAB	D14B-D155	Perform negation
DEAC-DED9	D156-D183	Constants
DEDA-DF2C	D184-D1D6	Perform EXP
DF2D-DF76	D1D7-D220	Series evaluation
DF77-DF7E	D221-D228	RND constants
DF7F-DFD7	D229-D281	Perform RND
DFD8-DFDE	D282-D288	Perform COS
DFDF-E027	D289-D2D1	Perform SIN
E028-E053	D2D2-D2FD	Perform TAN
E054-E08D	D2FE-D32B	Constants
E08C-E0BB	D32C-D35B	Perform ATN
E0BC-E0F8	D35C-D398	Constants
E0F9-E110	D399-D3B5	CHRGET sub for zero page
E116-E1B6	D3B6-D471	Basic cold start
E1B7-E1DD	D448-D471	Messages, BYTES FREE, ### COMMODORE BASIC ###
	D7AC-D802	Perform RECORD
	D803-D837	Disk parameter checks
	D838-D872	Dummy disk control messages
	D873-D919	Perform CATALOG or DIRECTORY
	D91A-D92E	Output

```

D92F-D941 Find spare secondary address
D942-D976 Perform DOPEN
D977-D990 Perform APPEND
D991-D9D1 Get disk status
D9D2-DA06 Perform HEADER
DA07-DA30 Perform DCLOSE
DA31-DA64 Set up disk record
DA65-DA7D Perform COLLECT
DA7E-DAA6 Perform BACKUP
DAA7-DAC6 Perform COPY
DAC7-DAD3 Perform CONCAT
DAD4-DB0C Insert command string values
DB0D-DB39 Perform DSAVE
DB3A-DB65 Perform DLOAD
DB66-DB98 Perform SCRATCH
DB99-DB9D Check Direct command
DB9E-DBD6 Query ARE YOU SURE?
DBD7-DBE0 Print BAD DISK
DBE1-DBF9 Clear DS$ and ST
DBFA-DC67 Assemble disk command string
DC68-DE29 Parse Basic DOS command
DE2C-DE48 Get Device number
DE49-DE86 Get File name
DE87-DE9C Get small variable parameter

** Entry points only for E000-E7FF **
E1DE E000 Register/screen initialization
E285 E0A7 Input from keyboard
E2F2 E116 Input from screen
E3D8 E202 Output character
E61B E442 Main interrupt entry
E62E E455 Interrupt: clock, cursor, keyboard
E6E4 E600 Exit from Interrupt
**
E76A-E7FF D717-D7AB MLM subroutines **
**
F000-F0B5 F000-F0D1 File messages
F0B6-F0B9 F0D2-F0D4 Send 'Talk'
F0BA-F0BB F0D5-F0D6 Send 'Listen'
F0BC-F0ED F0D7-F108 Send IEEE command character
F0EE-F127 F109-F142 Send byte to IEEE
F128-F135 F143-F150 Send byte and clear ATN
F151-F16B Option: timeout or wait
F136-F13F F16C-F16F DEVICE NOT PRESENT
F140-F155 F170-F184 Timeout on read, clear control lines
F156-F163 F185-F192 Send canned file message
F164-F16E F193-F19D Send byte, clear control lines
F16F-F17E F19E-F1AD Send normal (deferred) IEEE char
F17F-F18B F1AE-F1BF Drop IEEE device
F18C-F1D0 F1C0-F204 Input byte from IEEE
F1D1-F1E0 F205-F214 GET a byte
F1E1-F231 F215-F265 INPUT a byte
F232-F26D F266-F2A1 Output a byte
F26E-F283 F2A2-F2B5 Abort files
F284-F28C F2B6-F2C0 Restore default I/O devices
F28D-F2A8 F2C1-F2DC Find/setup file data
F2A9-F300 F2DD-F334 Perform CLOSE
F301-F30E F335-F342 Test STOP key
F30F-F314 F343-F348 Action STOP key
F315-F31C F349-F350 Send message if Direct mode
F31D-F321 F351-F355 Test if Direct mode
F322-F3C1 F356-F400 Program load subroutine
F3C2-F409 F401-F448 Perform LOAD
F40A-F42D F449-F46C Print SEARCHING
F42E-F43D F46D-F47C Print LOADING or VERIFYING
F43E-F45F F47D-F4A4 Get Load/Save parameters

F466-F493 F4A5-F4D2 Send name to IEEE
F494-F4B6 F4D3-F4F5 Find specific tape header
F4B7-F4CD F4F6-F50C Perform VERIFY
F4CE-F50D F50D-F55F Get Open/Close parameters
F521-F5A5 F560-F5E4 Perform OPEN
F5A6-F5D9 F5E5-F618 Find any tape header
F5DA-F63B F619-F67A Write tape header
F63C-F655 F67B-F694 Get start/end addrs from header
F656-F66B F695-F6AA Set buffer address
F66C-F683 F6AB-F6C2 Set buffer start & end addrs
F684-F68C F6C3-F6CB Perform SYS
F68D-F69D F6CC-F6DC Set tape write start & end
F69E-F728 F6DD-F767 Perform SAVE
F729-F76F F768-F7AE Update clock
F770-F7BB F7AF-F7FD Connect input device
F7BC-F805 F7FE-F84A Connect output device
F806-F811 F84B-F856 Bump tape buffer pointer
F812-F834 F857-F879 Wait for PLAY
F835-F846 F87A-F88B Test cassette switch
F847-F854 F88C-F899 Wait for RECORD
F855-F885 F89A-F8CA Initiate tape read
F886-F89A F8CB-F8DF Initiate tape write
F89B-F8E5 F8E0-F92A Common tape I/O
F8E6-F8EF F92B-F934 Test I/O complete
F8F0-F8FF F935-F944 Test STOP key
F900-F930 F945-F975 Tape bit timing adjust
F931-F9A6 F976-FA9B Read tape bits
FA57-FB75 FA9C-FBBA Read tape characters
FB76-FB7E FB8B-FBC3 Reset tape read address
FB7F-FB83 FBC4-FBC8 Flag error into ST
FB84-FB92 FBC9-FBD7 Reset counters for new byte
FB93-FBAE FBD8-FBF3 Write a bit to tape
FBAF-FC40 FBF4-FC85 Tape write
FC41-FC7A FC86-FCBF Write tape leader
FC7B-FC95 FCC0-FCDA Terminate tape; restore interrupt
FC96-FCA5 FCDB-FCEA Set interrupt vector
FCA6-FCB3 FCEB-FCF8 Turn off tape motor
FCB4-FCC5 FCF9-FD0A Checksum calculation
FCC6-FCD0 FD0B-FD15 Advance load/save pointer
FCD1-FCFD FD16-FD4B Power-on Reset
FD01-FD10 FD4C-FD5C Table of interrupt vectors
FD11-FFB0 D472-D716 Machine Language Monitor
** Jump table:
FF93-FF9E CONCAT,DOPEN,DCLOSE,RECORD
FF9F-FFAA HEADER,COLLECT,BACKUP,COPY
FFAB-FFB6 APPEND,DSAVE,DLOAD,CATALOG
FFB7-FFBC RENAME,SCRATCH
FFBD Get disk status
FFC0 OPEN
FFC3 CLOSE
FFC6 Set input device
FFC9 Set output device
FFCC Restore default I/O devices
FFCF INPUT a byte
FFD2 Output a byte
FFD5 LOAD
FFD8 SAVE
FFDB VERIFY
FFDE SYS
FFE1 Test stop key
FFE4 GET byte
FFE7 Abort all files
FFEA Update clock
FFFA-FFFF Hard vectors: NMI, Reset, INT

```

BASIC 4.0 Memory Map

Compiled by Jim Butterfield

There are some differences between usage between the 40- and 80-column machines.

Hex	Decimal	Description
0000-0002	0-2	USR jump
0003	3	Search character
0004	4	Scan-between-quotes flag
0005	5	Input buffer pointer; # of subscripts
0006	6	Default DIM flag
0007	7	Type: FF=string, 00=numeric
0008	8	Type: 80=integer, 00=floating point
0009	9	Flag: DATA scan; LIST quote; memory
000A	10	Subscript flag; FNX flag
000B	11	0=INPUT; \$40=GET; \$98=READ
000C	12	ATN sign/Comparison Evaluation flag
000D-000F	13-15	Disk status DS\$ descriptor
0010	16	Current I/O device for prompt-suppress
0011-0012	17-18	Integer value (for SYS, GOTO etc)
0013-0015	19-21	Pointers for descriptor stack
0016-001E	22-30	Descriptor stack(temp strings)
001F-0022	31-34	Utility pointer area
0023-0027	35-39	Product area for multiplication
0028-0029	40-41	Pointer: Start-of-Basic
002A-002B	42-43	Pointer: Start-of-Variables
002C-002D	44-45	Pointer: Start-of-Arrays
002E-002F	46-47	Pointer: End-of-Arrays
0030-0031	48-49	Pointer: String-storage(moving down)
0032-0033	50-51	Utility string pointer
0034-0035	52-53	Pointer: Limit-of-memory
0036-0037	54-55	Current Basic line number
0038-0039	56-57	Previous Basic line number
003A-003B	58-59	Pointer: Basic statement for CONT
003C-003D	60-61	Current DATA line number
003E-003F	62-63	Current DATA address
0040-0041	64-65	Input vector
0042-0043	66-67	Current variable name
0044-0045	68-69	Current variable address
0046-0047	70-71	Variable pointer for FOR/NEXT
0048-0049	72-73	Y-save; op-save; Basic pointer save
004A	74	Comparison symbol accumulator
004B-0050	75-80	Misc work area, pointers, etc
0051-0053	81-83	Jump vector for functions
0054-005D	84-93	Misc numeric work area
005E	94	Accum#1: Exponent
005F-0062	95-98	Accum#1: Mantissa
0063	99	Accum#1: Sign
0064	100	Series evaluation constant pointer
0065	101	Accum#1 hi-order (overflow)
0066-006B	102-107	Accum#2: Exponent, etc.
006C	108	Sign comparison, Acc#1 vs #2
006D	109	Accum#1 lo-order (rounding)
006E-006F	110-111	Cassette buff len/Series pointer
0070-0087	112-135	CHRGET subroutine; set Basic char
0077-0078	119-120	Basic pointer (within subrtn)
0088-008C	136-140	Random number seed.
008D-008F	141-143	Jiffy clock for TI and TI\$
0090-0091	144-145	Hardware interrupt vector
0092-0093	146-147	BRK interrupt vector
0094-0095	148-149	NMI interrupt vector
0096	150	Status word ST
0097	151	Which key down; 255=no key
0098	152	Shift key: 1 if depressed
0099-009A	153-154	Correction clock
009B	155	Keyswitch PIA: STOP and RVS flags
009C	156	Timing constant for tape
009D	157	Load=0, Verify=1
009E	158	Number of characters in keybd buffer
009F	159	Screen reverse flag
00A0	160	IEEE output; 255=character pending
00A1	161	End-of-line-for-input pointer
00A3-00A4	163-164	Cursor loc (row, column)
00A5	165	IEEE output buffer

00A6	166	Key image
00A7	167	0=flash cursor
00A8	168	Cursor timing countdown
00A9	169	Character under cursor
00AA	170	Cursor in blink phase
00AB	171	EOT received from tape
00AC	172	Input from screen/from keyboard
00AD	173	X save
00AE	174	How many open files
00AF	175	Input device, normally 0
00B0	176	Output CMD device, normally 3
00B1	177	Tape character parity
00B2	178	Byte received flag
00B3	179	Logical Address temporary save
00B4	180	Tape buffer character; MLM command
00B5	181	File name pointer; MLM flag, counter
00B7	183	Serial bit count
00B9	185	Cycle counter
00BA	186	Tape writer countdown
00BB-00BC	187-188	Tape buffer pointers, #1 and #2
00BD	189	Write leader count; read pass1/2
00BE	190	Write new byte; read error flag
00BF	191	Write start bit; read bit seq error
00C0-00C1	192-193	Error log pointers, pass1/2
00C2	194	0=Scan/1-15=Count/\$40=Load/\$80=End
00C3	195	Write leader length; read checksum
00C4-00C5	196-197	Pointer to screen line
00C6	198	Position of cursor on above line
00C7-00C8	199-200	Utility pointer: tape, scroll
00C9-00CA	201-202	Tape end addr/End of current program
00CB-00CC	203-204	Tape timing constants
00CD	205	0=direct cursor, else programmed
00CE	206	Tape read timer 1 enabled
00CF	207	EOT received from tape
00D0	208	Read character error
00D1	209	# characters in file name
00D2	210	Current file logical address
00D3	211	Current file secondary addr
00D4	212	Current file device number
00D5	213	Right-hand window or line margin
00D6-00D7	214-215	Pointer: Start of tape buffer
00D8	216	Line where cursor lives
00D9	217	Last key/checksum/misc.
00DA-00DB	218-219	File name pointer
00DC	220	Number of INSERTs outstanding
00DD	221	Write shift word/read character in
00DE	222	Tape blocks remaining to write/read
00DF	223	Serial word buffer
00E0-00F8	224-248	(40-column) Screen line wrap table
00E0-00E1	224-225	(80-column) Top, bottom of window
00E2	226	(80-column) Left window margin
00E3	227	(80-column) Limit of keybd buffer
00E4	228	(80-column) Key repeat flag
00E5	229	(80-column) Repeat countdown
00E6	230	(80-column) New key marker
00E7	231	(80-column) Chime time
00E8	232	(80-column) HOME count
00E9-00EA	233-234	(80-column) Input vector
00EB-00EC	235-236	(80-column) Output vector
00F9-00FA	249-250	Cassette status, #1 and #2
00FB-00FC	251-252	MLM pointer/Tape start address
00FD-00FE	253-254	MLM, DOS pointer, misc.
0100-010A	256-266	STR\$ work area, MLM work
0100-013E	256-318	Tape read error log
0100-01FF	256-511	Processor stack
0200-0250	512-592	MLM work area; Input buffer
0251-025A	593-602	File logical address table
025B-0264	603-612	File device number table
0265-026E	613-622	File secondary adds table
026F-0278	623-632	Keyboard input buffer
027A-0339	634-825	Tape#1 input buffer
033A-03F9	826-1017	Tape#2 input buffer
033A	826	DOS character pointer
033B	827	DOS drive 1 flag
033C	828	DOS drive 2 flag
033D	829	DOS length/write flag
033E	830	DOS syntax flag

daisy. The printer can print at speeds of up to 60 characters a second and produces output of great quality, especially if carbon ribbons are used. This newsletter is photographically reproduced from the output of a daisy-wheel printer and readers can compare the text with program listings printed on a dot matrix printer. Having the two printers would provide a backup system in the case of breakdowns, which becomes very attractive if you become, as is highly likely, increasingly reliant on the computer. An alternative is of course to hire a daisy-wheel when required.

People vary in their ability to read computer printout which has been produced on a dot-matrix printer. Some cannot adjust to the fact that the descending parts of letters do not drop below the line. If frequent users of the output suffer from this disability, then you may be forced to pay more for a printer with a more complex matrix head. The point is that the common head is of a design which requires characters to be designed out of 49 dots, arranged in a square, 7 X 7. This forces design compromises. Printers exist which use 9 X 7 or even more complex heads but these are correspondingly more expensive.

We should now consider the question of noise.

It is well worth making sure that the printer which sounds quiet at an exhibition is in fact acceptably quiet in the more peaceful office situation. You should also watch out for the amount of vibration and whether this is likely to cause disturbance. Acoustic covers to reduce noise are well worth considering. Thermal printers are very silent but the output has a tendency to fade when exposed to prolonged sunlight.

To turn to more mundane matters. The choice of appropriate paper should be looked at with some care. Prices for identical paper vary a lot and shopping around can pay handsome dividends.

Consider whether you will save valuable time using "one time carbon" or "no carbon required" two, three or four part sets, where multiple copies are required.

It is worth looking at various weights of paper, especially where word processing is involved.

It is possible to use ordinary tractor feed stationery for many purposes, but you may wish to produce a single sheet output for some work. In either case, you should look at the type of paper-feed offered by the printer of your choice, and decide if it fits your requirements. If not, find out how much extra it will cost to have the additional equipment fitted to, for instance, change the friction-fed daisy wheel to a tractor feed.

Surprisingly enough, it is even possible to have separate letter-heads glued to continuous paper, so that standard letters may be produced on a tractor fed printer and then detached this will then be indistinguishable from individual typewritten versions. This method is very expensive however.

It is worth looking at the type of ribbons available also. If you have the choice of fabric and carbon ribbons, you can make much better-looking printing available, when necessary. Some printers will even allow printing in red to occur, controlled by the program!

The conclusion of all this is that as much care should be given to printer selection as to the selection of the computer itself. Surveys are published from time to time in the commercial magazines and you should study these in depth. As a last bit of advice make sure you are quoted prices with the printer interface included as this can cost over £100 pounds and ensure that ribbons and service are available.

Cursor — Tape Magazine for the PET

Cursor, the bi-monthly cassette magazine is now available from Audiogenics Ltd, 34 Crown Street, Reading, Berkshire (tel: 0734 595269). The price is £3.50 + 25p p&p, or £21 (including p&p) for six issues. Back issues are also available by request to Martin Maynard.

033F-0340	831-832	DOS disk ID
0341	833	DOS command string count
0342-0352	834-850	DOS file name buffer
0353-0380	851-896	DOS command string buffer
03EE-03F7	1006-1015	(80-column) Tab stop table
03FA-03FB	1018-1019	Monitor extension vector
03FC	1020	IEEE timeout defeat
0400-7FFF	1024-32767	Available RAM including expansion
8000-83FF	32768-33791	(40-column) Video RAM
8000-87FF	32768-34815	(80-column) Video RAM
9000-AFFF	36864-45055	Available ROM expansion area
B000-DFFF	45056-57343	Basic, DOS, Machine Language Monitor
E000-E7FF	57344-59391	Screen, Keyboard, Interrupt programs
E810-E813	59408-59411	PIA 1 - Keyboard I/O
E820-E823	59424-59427	PIA 2 - IEEE-488 I/O
E840-E84F	59456-59471	VIA - I/O and timers
E880-E881	59520-59521	(80-column) CRT Controller
F000-FFFF	61440-65535	Reset, I/O handlers, Tape routines



Digital Design and Development

18/19 Warren Street · London W1P 5DB Tel: 01 387 7388

CBM PET

Specialist Suppliers of Complete Systems for Industrial and Laboratory Monitoring and Control.

**Please note our new address.
Callers welcomed for demonstration
and/or discussion.**



PET INTERFACES

IEEE-488 Compatible Units

- | | |
|-------------------------------------|--------|
| ● 16 Channel 8-Bit A/D Converter | £300 |
| ● 8 Channel 8-Bit D/A Converter | £350 |
| ● 8 Channel 12-Bit A/D Converter | £600 |
| ● 12-Bit D/A Converter | P.O.A. |
| ● X-Y Analog Plotter Interface | £200 |
| ● Digital Data Input Unit, 64 Bits | £400 |
| ● Digital Data Output Unit, 64 Bits | £350 |
| ● 16 Channel Relay Unit | £350 |

Also....

- | | |
|--|--------|
| ● USER Port Converter A/D plus D/A | £200 |
| ● Fast Data Acquisition System
40,000 readings per sec. 4 A/D + 4 D/A | P.O.A. |

All units boxed complete with IEEE-488 address internally selectable, with integral power supply, cables, switch, fuse, indicators and illustrative BASIC software.

TERMS: All prices EX-VAT. P&P extra.
Cheques should be made payable to
3D Digital Design & Development.
All goods supplied under 90 days warranty.
CUSTOM DESIGN UNDERTAKEN

